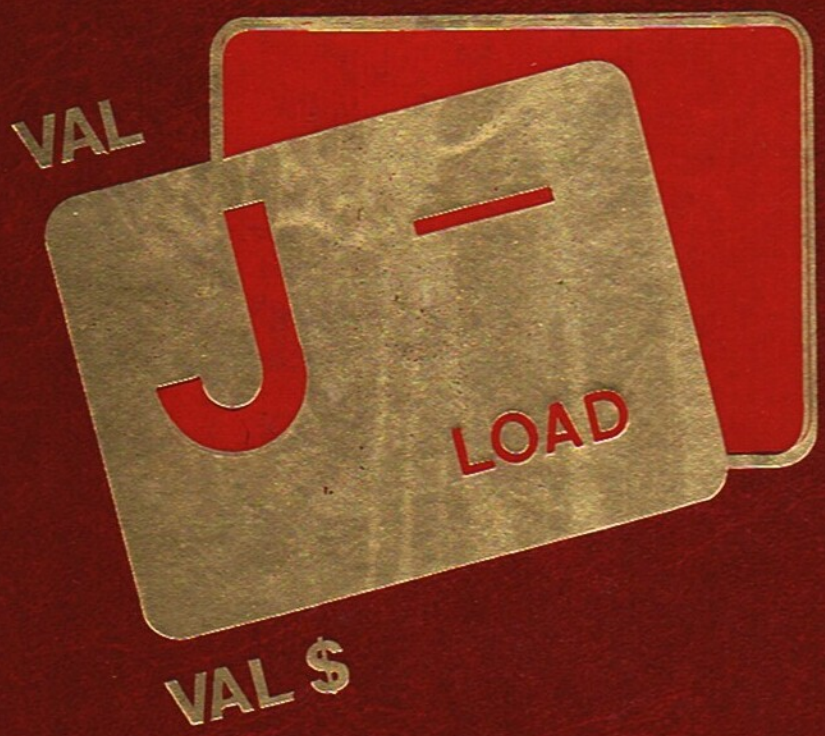
RUN

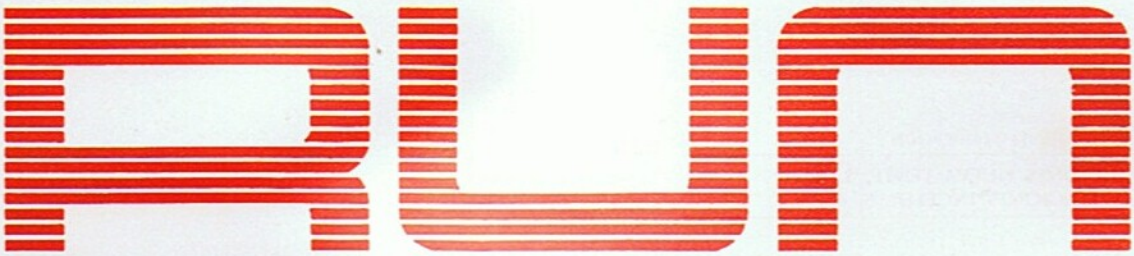
Enciclopedia Práctica del Spectrum



Nueva Lente/Ingelek

RUN

Enciclopedia Práctica del Spectrum



Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

Una publicación

EDICIONES NUEVA LENTE, S. A.
y EDICIONES INGELEK, S. A.

Director-editor por Nueva Lente, S. A.:
MIGUEL J. GONI
Director-editor por Ingelek, S. A.:
ANTONIO M. FERRER
Director de producción:
RICARDO ESPAÑOL
Jefe de producción:
SANTOS ROBLES

Director de la obra:
FERNANDO LOPEZ MARTINEZ
Coordinación:
VICENTE ROBLES
Colaboradores:
JUAN MANUEL LOPEZ MARTINEZ
CARLOS DE LA OSSA
JUAN MANUEL MAYORAL

Diseño Gráfico:
JOSE OCHOA
Maquetación:
CARLOS GONZALEZ-AMEZUA
Fotografía:
«Equipo Gálata»
(ALBINO LOPEZ y EDUARDO AGUDELO)
Ilustraciones:
JOSE OCHOA
ALFONSO MENDEZ
ANTONIO PERERA

© Ediciones Nueva Lente, S. A.
Madrid 1985
Dirección, redacción y administración:
Benito Castro, 12. 28028-Madrid
Teléfs.: 245 45 98 y 246 73 67

© Ediciones Ingelek, S. A.
Madrid 1985
Dirección, redacción y administración:
Avda. Alfonso XIII, 141. 28016-Madrid
Teléf.: 250 58 20

ISBN tomo primero: 84-7534-120-9
ISBN tomo segundo: 84-7534-121-7
ISBN tomo tercero: 84-7534-122-5
ISBN tomo cuarto: 84-7534-123-3
ISBN de la obra: 84-7534-118-7

Fotomecánica: OCHOA, S. A.
Ricardo Ortiz, 74. 28017-Madrid

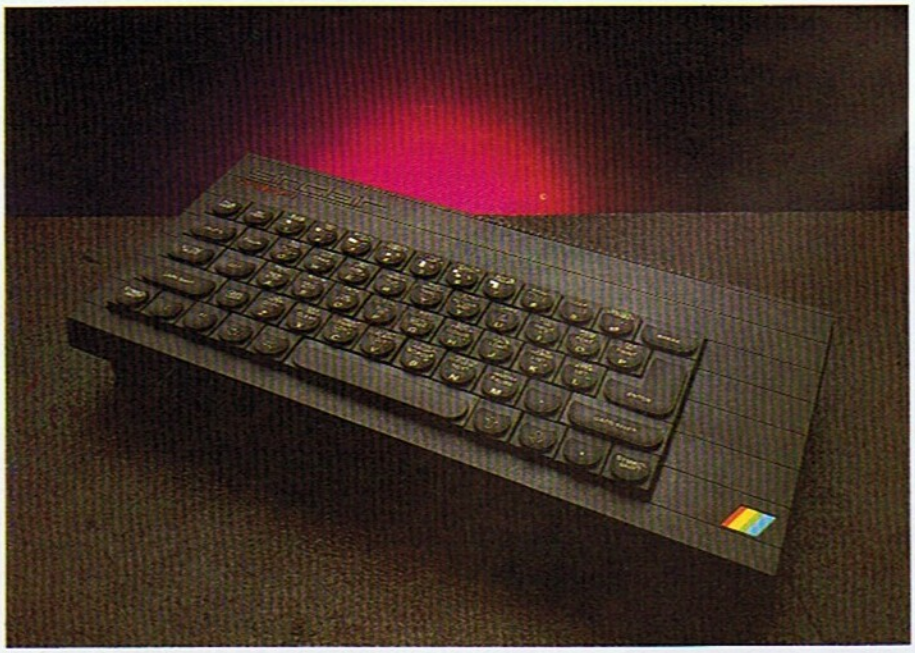
Impresión: GRAFICAS REUNIDAS, S. A.
Avda. de Aragón, 56. 28027-Madrid

Depósito legal: M-9896-1985
PRINTED IN SPAIN

Queda prohibida la reproducción total o parcial de esta obra sin permiso escrito de los editores.

El Sinclair ZX Spectrum es, sin lugar a dudas, la estrella más rutilante del firmamento informático de nuestro país. Desde su aparición en el mercado nacional, allá por el año 1983 su popularidad ha ido creciendo imparablemente, hasta alcanzar cotas casi inimaginables. Ríos de tinta se han vertido sobre este pequeño aparato, y una pléyade de revistas, libros y programas, ha invadido el mercado microinformático con llamativos anuncios y sugerentes ofertas. Sin embargo, hasta la aparición de RUN Enciclopedia Práctica del Spectrum, ninguna publicación se había encargado de ser un eficaz y un paciente tutor para el aprendizaje del manejo de este potente aparato.

Gracias a ella, no nos vamos a ver obligados a introducir pasivamente en nuestro ordenador los programas diseñados por otros, sino que estaremos perfectamente capacitados para ejercer una actitud crítica, que nos permita mejorar los programas ajenos y crear los nuestros propios. Y lo que es más, nos hará posible alcanzar el nivel de conocimientos suficientes como para poder lanzarnos al aprendizaje de otros lenguajes, en principio tan arduos, como el código máquina.



Para sacar todo el provecho de un aparato tan completo como el Spectrum, no basta una simple guía del usuario, es necesario mucho más:
— Un curso de programación eficaz, que orientado de una manera altamente pedagógica consiga enseñar sin aburrir.
— Un conocimiento sobre los trucos y técnicas, que sólo adquiriríamos con muchos meses de experiencia en la programación.
— Una orientación clara sobre los complementos del equipo que se encuentra a nuestro alcance.
— Y como no, una parte de entretenimiento, que nos permita disfrutar y practicar al mismo tiempo en nuestro aparato, mediante divertidos y útiles programas.
Todo esto es RUN Enciclopedia Práctica del Spectrum. Para cumplir plenamente su objetivo, la obra se ha dividido en tres secciones:

BASIC: *un completísimo curso de BASIC, que nos permitirá conocer a la perfección el lenguaje de programación de nuestro ordenador. Desde sus más simples fundamentos, hasta las complicadas variables del sistema.*

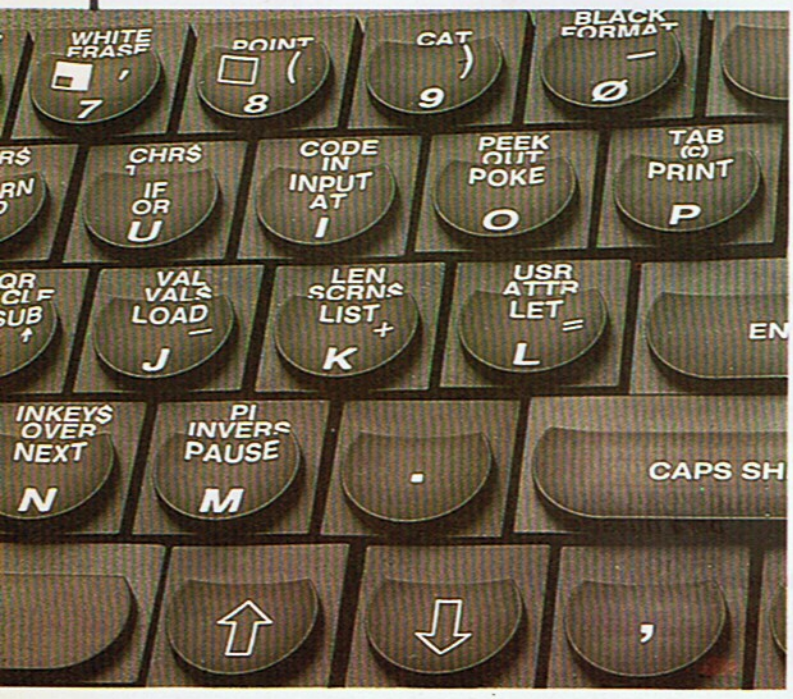
TU SPECTRUM: *la sección en la cual profundizaremos en las peculiares características del ordenador, tanto en la programación, a través de trucos y consejos, como en su aspecto físico, constituyéndose en una completa guía para navegar por el mar de complementos existentes para nuestro Spectrum.*

PROGRAMAS: *por supuesto, no todo va a ser teoría y en esta sección podremos divertirnos todos, con programas de la más diversa índole que abarcan desde el juego hasta la ayuda a la programación.*

BASIC

El lenguaje de programación del Spectrum; nuestro medio de comunicación con él, y por tanto, un punto neurálgico en el aprendizaje de la programación del ordenador.
A lo largo de la obra, un desarrollo progresivo y pedagógico nos permitirá llegar a conocer al máximo el dialecto BASIC Sinclair; capacitándonos para la realización de cualquier programa en BASIC, por complicado que éste parezca.

Con la ayuda de gran cantidad de explicaciones, gráficos y ejemplos, iremos adentrándonos poco a poco en las interioridades del lenguaje de programación más difundido en el mundo. No sólo algo necesario para nuestro presente, sino imprescindible para el futuro.



Comenzaremos por aprender a comunicarnos con el ordenador, mediante el correcto uso del teclado; a partir de este punto, iremos avanzando semana a semana, hasta completar una primera fase de aprendizaje en la que ya dominaremos la introducción de cualquier palabra BASIC, sabremos utilizar nuestro Spectrum como una supercalculadora o introducir y corregir cualquier programa BASIC.

En una segunda etapa, comenzaremos a desvelar los secretos de la programación básica de nuestro aparato, que sentarán los más sólidos cimientos para la construcción de unos programas de calidad.

Según vayamos avanzando en la obra, comprobaremos como nuestra capacidad para la realización de espectaculares programas, crece por momentos. Dominaremos ya la técnica de la programación y así podremos comenzar a dominar también el arte de la programación. Utilizaremos a pleno rendimiento el sonido, los gráficos, el color, los caracteres redefinibles...

Finalmente, llegaremos al punto álgido de nuestra fase de aprendizaje. Abriremos una puerta a la ampliación de conocimientos, mediante un primer contacto con otros lenguajes utilizables en el Spectrum, especialmente el temido código máquina. Conoceremos también técnicas de programación general, aplicables a cualquier lenguaje, que nos permitirán crear programas eficaces con el menos esfuerzo.

En definitiva, podremos afirmar que dominamos el lenguaje de nuestro ordenador; y no sólo éso, sino también los fundamentos de microinformática que nos permitan abrirnos camino en este mundo apasionante.

TU SPECTRUM

Aprovechar al máximo la capacidad de nuestro ordenador, no se limita a conocer perfectamente su lenguaje de programación. Es indispensable también tener un amplio conocimiento del aparato en sí, aprendiendo los mil y un trucos que convierten a un simple programador, en un buen programador.

Nuestro Spectrum es un aparato con una increible capacidad de expansión. Disponemos de multitud de complementos con los que ampliar las facultades del equipo básico: desde simples mandos de juego, hasta complicados interfaces. Por todo ello, es necesaria una guía que nos informe imparcialmente sobre las características de estos aparatos; sus ventajas e inconvenientes, y por supuesto, su forma de manejo.
En esta sección se van a tratar por tanto tres puntos fundamentales:

- Pequeños trucos, de fácil incorporación en cualquier programa, que permiten extraer un máximo provecho de las características del Spectrum.
Algunos de ellos se encaminarán hacia la utilidad más concreta, en cuanto a la facilidad en la programación. Así, podremos aprender a grabar programas BASIC en forma de código máquina, o aprovechar rutinas existentes en ROM mediante simples llamadas al código máquina, aunque no conozcamos nada sobre este lenguaje.
Otros se dedicarán a distintos aspectos, pero también importantes, como son los efectos de gran espectacularidad que asombrarán a los usuarios de nuestros programas: rápidos cambios de pantallas, utilización de varios colores en el borde de la pantalla, gráficos de alta resolución...

- Exposición de temas de gran interés para el usuario del Spectrum, acompañados de ejemplos prácticos que nos permitan construir una biblioteca de subrutinas, empleadas por profesionales de la programación, que nosotros también podremos utilizar en nuestros propios programas.
Así seremos capaces de controlar las zonas de memoria RAM a nuestro antojo, conoceremos los sistemas empleados para protección del software, y seremos capaces de construir bases de datos, para la programación de espectaculares juegos multi-pantalla.

- Revisión de los complementos que existen para nuestro aparato. Con exhaustivas explicaciones sobre sus características, facilidad de manejo, utilidad, etc...

Esta última misión de la sección TU SPECTRUM, es convertirse en una guía del comprador, donde podremos encontrar toda la información necesaria sobre impresoras, joysticks, unidades de almacenamiento de programas, ampliaciones de memoria y otros complementos excepcionalmente especiales, como el adaptador de teclado musical o el lápiz óptico.

PROGRAMA

Todas las semanas, podremos disfrutar de la diversión y utilidad que proporciona nuestro Spectrum, gracias al programa que aparecerá en la última sección.
Acompañado de un comentario explicativo, nos permitirá también ir aprendiendo algo más sobre la programación.
La temática de los programas será muy variada, aunque fundamentalmente podremos dividirlos en dos bloques:

- Programas de juego.
- Programas de utilidad.

Dentro del primer bloque, se presentarán muy diversos programas destinados al entretenimiento, incluyendo juegos, tanto de «tablero», como de acción.

Dentro de los primeros habrá para todos los gustos, tratándose la vertiente de la mera distracción, con juegos tradicionales, como el de la «Batalla Naval» y la del juego de inteligencia o rapidez mental, sin descuidar en ningún momento su amenidad.

Los juegos de acción nos transportarán a los más diversos mundos, y nos situarán en las más difíciles situaciones: el granjero de Marte que se defiende de las tarántulas supervitaminadas, o el arriesgado piloto espacial, que debe surcar los peligrosos anillos de Saturno, evitando el contacto fatal con los asteroides hacia los que se precipita.
En cuanto a los programas de utilidad, sus objetivos serán de muy diverso tipo; irán desde los educativos más entretenidos, hasta las ayudas a la realización de programas.

Los de este último tipo, podrán ser incluidos como subrutinas en nuestros propios programas, permitiéndonos construir una «caja de herramientas» en la que poder apoyarnos en cualquier momento para la realización de nuestro software particular.

Por otra parte, todos los programas publicados, sean del tipo que sean, podrán servir siempre como orientación sobre la forma en que se pueden resolver determinados problemas de programación, estudiando la manera en que un profesional ha llevado a cabo este trabajo.

Finalmente, para evitar el esfuerzo de la introducción de algunos programas de gran longitud, unas casetes de alta calidad, conteniendo el software aparecido hasta el momento en la obra, serán puestas a la venta de forma periódica.

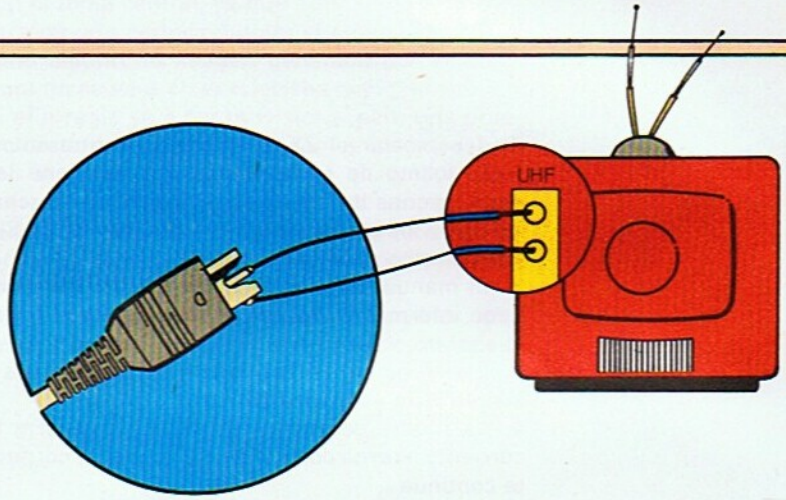
EL SPECTRUM

UN nuevo amigo ha llegado a casa, capaz no sólo de ser nuestro compañero en el ocio, sino también en el trabajo.

Deseoso de estar con nosotros y ayudarnos en lo que pueda. Pero sólo tiene un defecto: no habla nuestro idioma. ¿Vamos a asustarnos ante tan pequeño inconveniente? No. ¡Animo! Su idioma es muy fácil de aprender, y casi sin darnos cuenta conseguiremos trabar amistad con el recién llegado, **el ordenador.**

Una de las características de nuestro electrónico amigo es que necesita más de la electricidad para vivir que nosotros de la comida, y por eso, cada vez que lo desenchufamos de la red, se sume en un sueño del que nosotros deberemos aprender a despertarlo.

Este será sin duda el primer paso: como hacer funcionar el ordenador. Pero antes vamos a echar un pequeño vistazo a su equipaje, a ver si se ha dejado algo en el aeropuerto.



Mediante dos simples, soldaduras podemos adaptar el conector macho de antena estándar, a las entradas dobles para UHF de algunos televisores.

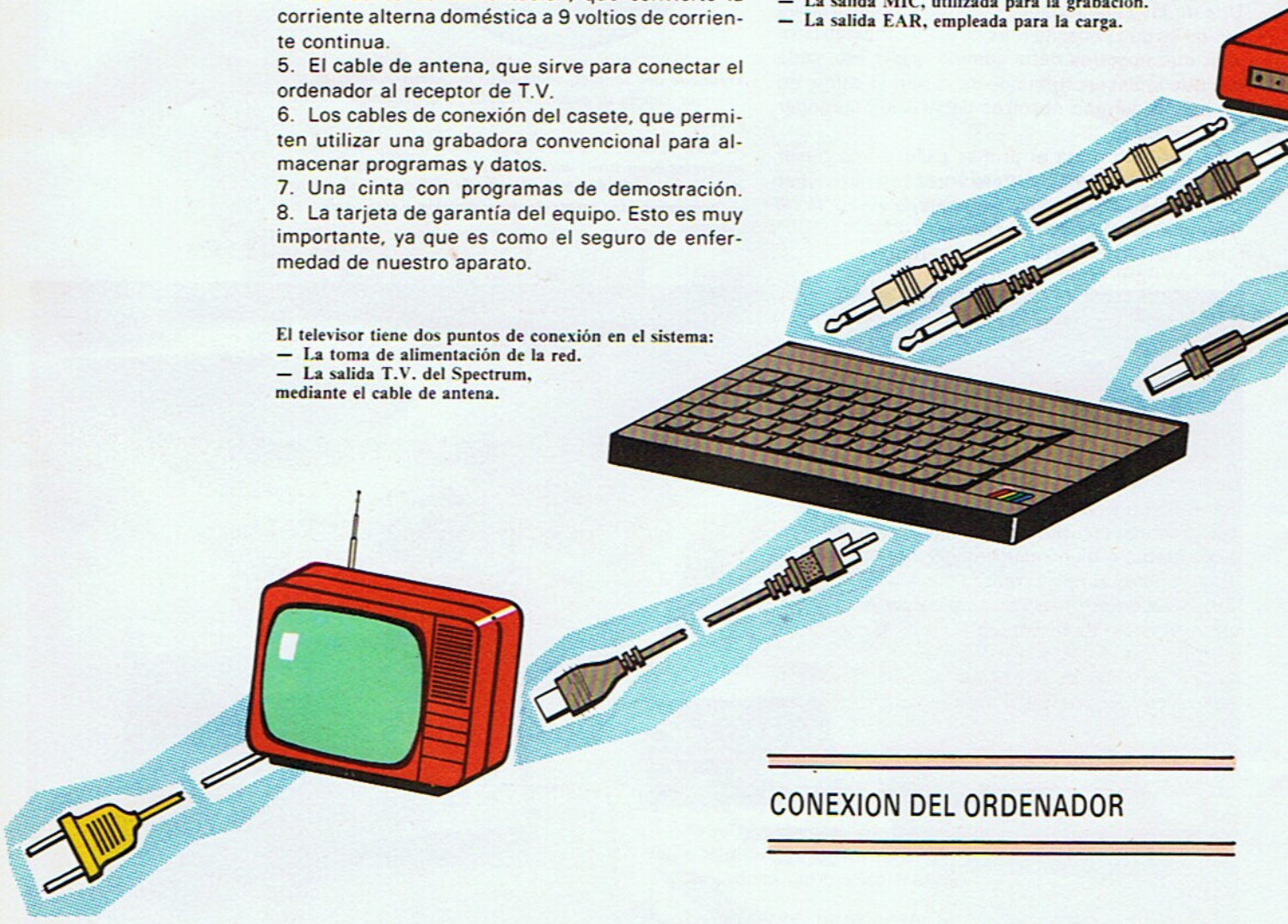
Al abrir la caja de nuestro Spectrum, encontraremos los siguientes elementos: unidad central (con el teclado), cable de antena, cable de casete, fuente de alimentación, manual de programación, casete de demostración y tarjeta de garantía.

COMPONENTES DEL EQUIPO

Al desembalar el ZX Spectrum encontraremos:
1. El folleto de introducción, que contiene las instrucciones de conexión y puesta en marcha, así como los principios fundamentales del BASIC y el correcto manejo del teclado.
2. El manual de programación BASIC, que contiene información acerca del lenguaje que habla el aparato.
3. El ordenador, en el que lo que más destaca a simple vista es el teclado.
4. La fuente de alimentación, que convierte la corriente alterna doméstica a 9 voltios de corriente continua.
5. El cable de antena, que sirve para conectar el ordenador al receptor de T.V.
6. Los cables de conexión del casete, que permiten utilizar una grabadora convencional para almacenar programas y datos.
7. Una cinta con programas de demostración.
8. La tarjeta de garantía del equipo. Esto es muy importante, ya que es como el seguro de enfermedad de nuestro aparato.

También va a ser necesario un receptor de T.V., en blanco y negro o en color, que servirá, entre otras cosas, para que podamos leer las respuestas a nuestras preguntas.
Es indiferente el empleo de un tipo de T.V. u otro. En el caso del B/N se obtendrán los colores blanco y negro y seis tonalidades de grises intermedios. En Color se obtendrán además del blanco y negro los colores azul oscuro, rojo, rosa oscuro, verde, azul claro y amarillo.
Una vez revisados los componentes del equipo, ya podemos empezar a dar los pasos para la instalación del aparato.

El periférico fundamental de almacenamiento de datos/programas es el casete. Sus puntos de conexión son:
— La salida MIC, utilizada para la grabación.
— La salida EAR, empleada para la carga.

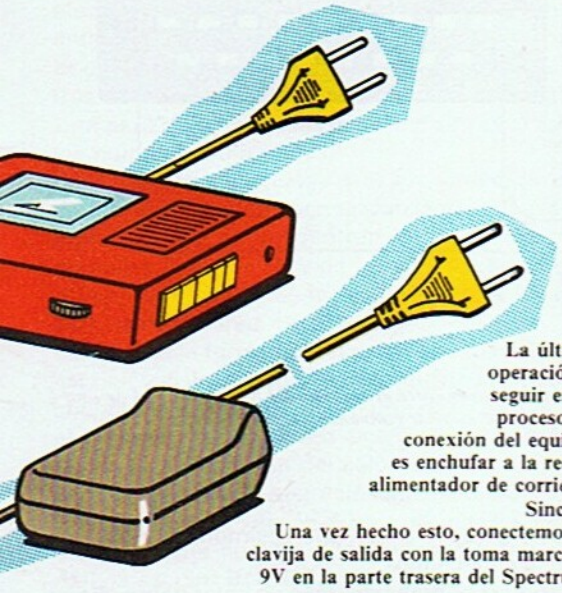


El televisor tiene dos puntos de conexión en el sistema:
— La toma de alimentación de la red.
— La salida T.V. del Spectrum, mediante el cable de antena.

CONEXION DEL ORDENADOR

En primer lugar, encendemos el receptor de T.V. colocando el volumen al mínimo. A continuación,

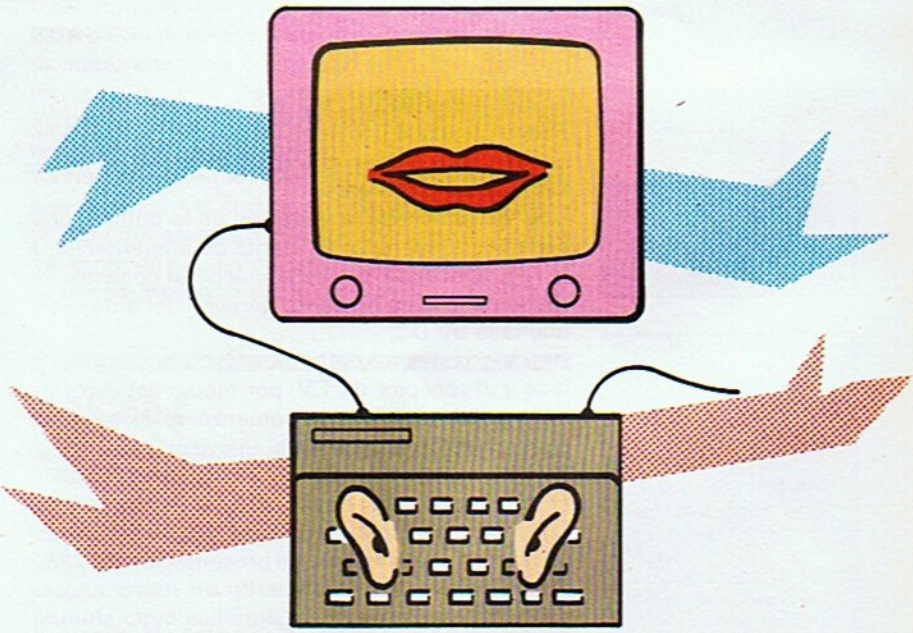
enchufamos el cable de antena al conector, situado en la parte posterior izquierda del ordenador (marcado T.V.) y colocamos el otro extremo del cable en la toma de antena del televisor. En el caso de tener dos entradas de antena diferentes, una para VHF y otra para UHF, lo conectaremos a esta última.



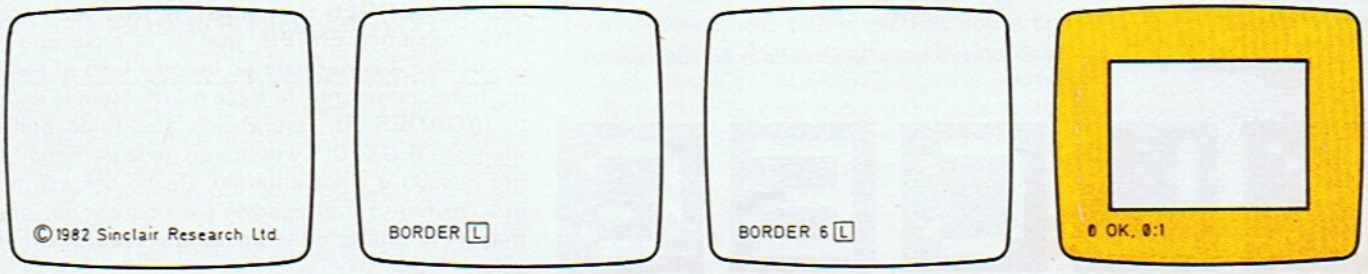
La última operación a seguir en el proceso de conexión del equipo, es enchufar a la red el alimentador de corriente Sinclair
Una vez hecho esto, conectemos la clavija de salida con la toma marcada 9V en la parte trasera del Spectrum.

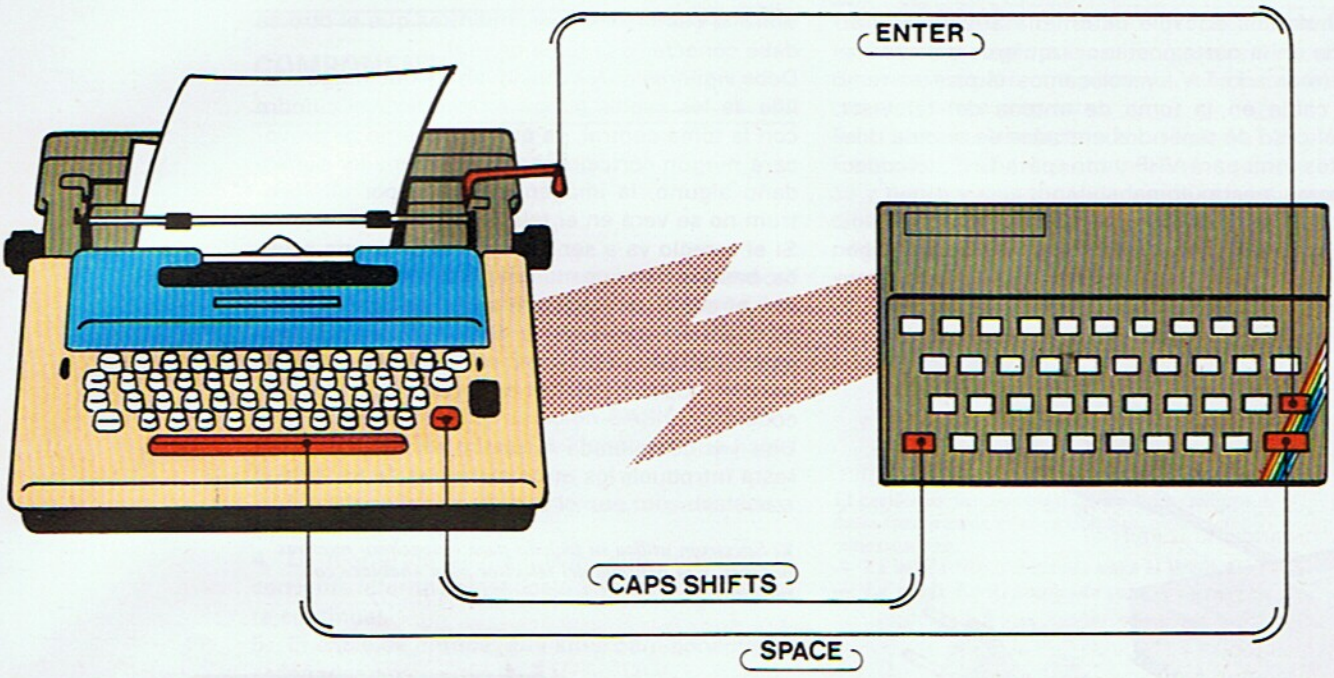
En algunos televisores antiguos, la clavija del cable de antena no tendrá el mismo formato que la entrada de la T.V. Para ser exactos, el tipo de toma antigua está compuesto por dos clavijas, en vez de una sola doble. Para solucionar esto, podemos recurrir a dos sistemas: o bien nos proveemos del adaptador conveniente en una tienda especializada, o bien recurrimos a un remedio casero con dos simples trozos de cable.
Como podemos observar, el conector en que termina el cable de la antena tiene la forma de un cilindro con una toma en el centro.
La operación debe comenzar con el «pelado» de dos cables por sus dos extremos; uno de ellos se unirá al cilindro exterior, mientras que el otro se debe conectar a la toma central.
Debe vigilarse muy especialmente que ningún hilillo de los cables ponga en contacto el cilindro con la toma central, ya que aunque no se provocará ningún cortocircuito ni el ordenador sufrirá daño alguno, la imagen generada por el Spectrum no se verá en el televisor.
Si el arreglo va a ser provisional para una prueba, bastará con que mantengamos los cables unidos al conector mediante cinta aislante o cualquier otra cinta adhesiva. Si por el contrario, deseamos crear un artilugio más duradero, lo mejor que podemos hacer es soldar los cables al conector.
Una vez completada la operación anterior, sólo resta introducir los otros extremos de los cables

El Spectrum utiliza su teclado para «escuchar» nuestras órdenes, y la pantalla del televisor para «hablar» con nosotros.



Secuencia de pantallas para la obtención de BORDER 6.





Las semejanzas entre el teclado del Spectrum y el de una máquina de escribir convencional, son muchas.

en las entradas de UHF del televisor. Para una buena conexión conviene utilizar las clavijas adecuadas para dichas entradas. El orden de conexionado es indiferente.

Una vez concluida la conexión de la antena, procedemos a enchufar la fuente de alimentación a la red, aplicando la clavija macho al conector situado en la zona posterior derecha del ordenador marcada 9V DC.

Una vez conectado el ordenador a la red por un lado y al receptor de T.V. por medio del cable de antena por otro, puede comenzarse la labor de sintonizar correctamente el televisor.

Esta operación consiste en ir girando la ruedecita del sintonizador hasta apreciar la imagen del Spectrum. La pantalla deberá presentar en su zona inferior el mensaje de presentación *(C) 1982 Sinclair Research Ltd.,* escrito en letras negras sobre fondo blanco. Para aquellos cuyo sintonizador tiene marcados los canales, aclararemos que el Spectrum es una pequeña emisora de T.V. que utiliza el canal 36 de UHF.

Si nuestra T.V. dispone de varios selectores de canal, lo ideal sería dejar uno destinado al ordenador, para no tener que resintonizar cada vez que lo conectemos.

Cuando la pantalla aparece de color negro, o con cualquier otra cosa diferente del mensaje de presentación, es señal de un mal encendido del ordenador, por lo que se debe desconectar la clavija de la fuente de alimentación y volver a conectarla pasados unos segundos.

Aquellos que posean un televisor en color, deberán comprobar, además de la nitidez de imagen, que el color se ve correctamente. Para ello, pulsamos la tecla **B**; esto hará que el mensaje de presentación se vea sustituido en la pantalla por la palabra **BORDER**, seguida de una **L** en negativo, que se enciende y apaga intermitentemente. Este carácter **L** lo denominamos **CURSOR**, pero ya hablaremos de él más adelante. Ahora vamos a terminar de comprobar la recepción del color. A continuación pulsamos la tecla **6**, con lo cual se desplazará el cursor apareciendo a continuación de **BORDER** un número 6. Una vez hecho esto, pulsamos **ENTER**, que es la tecla que se encuentra a la derecha de la letra **L** en el teclado. Inmediatamente, la frase que habíamos escrito **(BORDER 6)**, habrá sido sustituída por el mensaje **0 OK, 0:1** y el marco de la pantalla habrá pasado a color amarillo. De no ser así, hay que resintonizar el aparato para conseguir, además de la nitidez de imagen, la visualización del color.

Forma de representación en pantalla de los cinco modos del cursor.

EL TECLADO

El Spectrum no puede oír y por tanto no podemos decirle las cosas de palabra; sin embargo posee una gran habilidad a la hora de leer y escribir. Por ello, se ha elegido el lenguaje escrito como forma de comunicación con el ordenador. A tal efecto, el aparato dispone de dos órganos fundamentales de comunicación: uno de entrada, que es el teclado y realiza una función similar a nuestros oídos, y otro de salida, la pantalla, que utilizará para decirnos cosas de forma similar a como los humanos utilizamos el habla.

Básicamente, el teclado del Spectrum es como el de una máquina de escribir corriente. Consta de números, que se encuentran en la fila superior de teclas (del 1 al 0), y de letras y símbolos que ocupan el resto de las teclas, además de las que cumplen funciones especiales.

La diferencia fundamental entre una máquina de escribir y el ordenador, es la cantidad de palabras que aparecen escritas sobre una misma tecla. En el proceso que seguimos antes para comprobar la correcta recepción del color, observamos que al pulsar la tecla **b**, apareció la palabra **BORDER** y no la letra **b**. ¿Por qué ocurrió ésto? Simplemente porque nuestro Spectrum sabía que queríamos escribir **BORDER** y no **b**.

Efectivamente. A la hora de utilizar el teclado, contaremos con una importante ayuda: el propio ordenador, que intentará averiguar qué es lo que queremos escribir.

Uno de los métodos por el cual el ordenador nos ayuda a utilizar el teclado, es mediante el cursor. Como ya dijimos anteriormente, la función del cursor es señalar el lugar por el cual vamos escribiendo, pero en el caso del Spectrum tiene aún una misión más: indicarnos qué tipo de palabra o letra está esperando el ordenador. Esto lo consigue utilizando para el cursor formas diferentes. Hasta ahora habíamos visto que utilizaba una **L** como cursor. Pasemos entonces a ver el resto de las formas que puede adoptar.

Apaguemos y encendamos el ordenador, y cuando aparezca el mensaje de presentación, pulsemos la tecla **ENTER**. Como se puede ver, el cursor no ha adoptado la forma de una **L**, sino de una **K**. Ahora, pulsemos a un mismo tiempo las teclas **CAPS SHIFT** y **SYMBOL SHIFT** y veremos como el cursor se transforma en una **E**. Repitiendo esta última operación, el cursor volverá a tomar su forma de **K**. Pero aún no han terminado las transformaciones. Pulsemos las teclas **CAPS SHIFT** y **9** a un tiempo; esta vez, la **K** se habrá visto sustituida por una **G** y pulsando de nuevo **CAPS SHIFT** y **9**, obtendremos la **K** inicial.

Las PALABRAS CLAVE aparecen serigrafiadas en blanco sobre las teclas alfabéticas.

El siguiente paso no es la primera vez que lo damos. Pulsamos la tecla **B**; la conocida palabra **BORDER** aparecerá seguida de un cursor **L**. Para terminar con esta demostración de las diversas formas que puede adoptar un cursor, pulsamos a un tiempo las teclas **CAPS SHIFT** y **2**. Una **C** habrá sustituido a la **L** anterior.

Ya hemos visto todas las formas diferentes que puede adoptar el cursor, que de ahora en adelante denominaremos MODOS del cursor. Demos por tanto un pequeño repaso. Los 5 modos en que se puede presentar el cursor son: **K, L, C, E** y **G**.

EL MODO K

El cursor, en este modo, indica que cualquier tecla correspondiente a una letra que se pulse, no será interpretada como esa letra, sino como la palabra que aparece escrita sobre la tecla.

Así, si encontrándonos en el modo **K** pulsamos una **A**, se escribirá la palabra **NEW**; si pulsamos una **R**, **RUN**, etc.

Estas palabras que están escritas sobre las teclas ALFABETICAS (teclas de arriba), se denominan en BASIC, PALABRAS CLAVE (en inglés *KEYWORDS)*. El ordenador sabe que cualquier instrucción que le demos, debe comenzar por una palabra clave o por un número (eso ya lo veremos más adelante); por eso, al comienzo de una línea siempre espera una de ellas, situándose en el modo **K**.

Las teclas en el Spectrum se dividen en tres tipos fundamentales:
1. Teclas alfabéticas.
2. Teclas numéricas.
3. Teclas de control.

Las teclas de control no producen la representación de un nuevo carácter en la pantalla al ser pulsadas; las numéricas y las alfabéticas sí.

Al no existir mayúsculas para las teclas numéricas, la pulsación de éstas en combinación con **CAPS SHIFT** las convertirá en teclas de control.

Emprendemos la labor detectivesca, para averiguar el significado del misterioso mensaje secreto, que ocultan las teclas del Spectrum.

Este sistema automático, nos ayudará a evitar el error de no empezar una orden por una palabra clave.

Como ya nos dimos cuenta cuando sintonizamos el color, al pulsar la tecla **b**, apareció la palabra **BORDER**, a pesar de que no había ningún tipo de cursor. Esto se debe a que, cuando el Spectrum escribe algún mensaje en la parte inferior de la pantalla, como por ejemplo el de presentación, se encuenta en modo **K**, aunque el cursor no aparezca para demostrarlo.

Para cerciorarnos de que nos encontramos en modo **K**, siempre que aparezca uno de estos mensajes podemos pulsar la tecla **ENTER**, y el mensaje que hubiera se verá sustituido por el cursor **K**.

Hemos visto que es lo que ocurre cuando se pulsa una tecla alfabética, pero ¿qué sucede cuando se pulsa una tecla NUMERICA? Simplemente, que aparecerá el número correspondiente, tal cual. Tengamos en cuenta que en las teclas de los números no aparece escrita ninguna palabra clave y, por tanto, el modo **K** no afecta de manera especial a estas teclas.

I. La solución al problema del mensaje secreto es:

REM	E
NEXT	N
GOSUB	H
POKE	O
RUN	R
NEW	A
BORDER	B
IF	U
REM	E
NEXT	N
NEW	A

ENHORABUENA.

Antes de proseguir nuestra visita por el teclado, vamos a realizar un entretenido ejercicio. Descifrar el mensaje que el ordenador propondrá mediante sus palabras clave. A continuación aparece una lista de palabras clave, cada una de las cuales corresponde a una letra del mensaje secreto. ¿Cuál será esa letra? Aquella que se encuentre en la misma tecla que la palabra clave. Ahí va una pista. La primera palabra clave del mensaje es **REM**, luego entonces la primera letra del mensaje oculto será **E**, puesto que en la tecla de la **E** es donde se obtiene la palabra clave **REM**. Ahora, el mensaje completo. Pero no hagamos trampa. No miremos la solución en el cuadro antes de terminar.

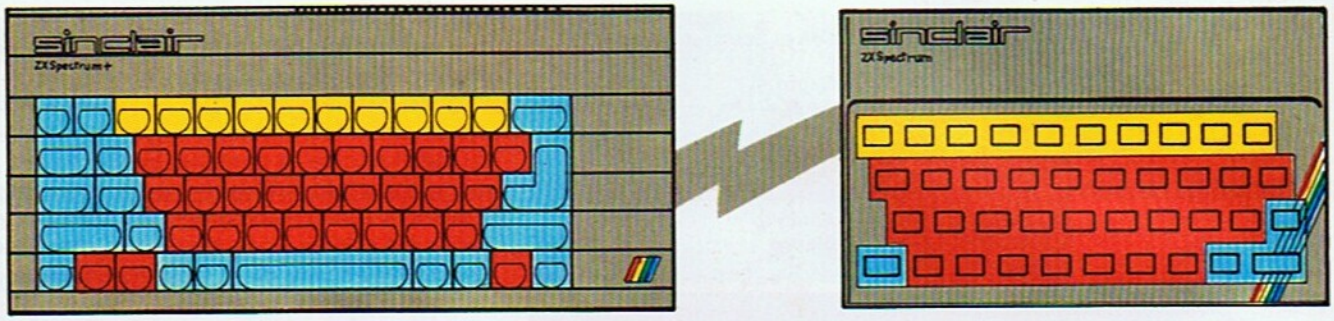
REM - NEXT - GOSUB - POKE - RUN - NEW - BORDER - IF - REM - NEXT - NEW.

EL MODO L

Como ya hemos visto, cuando el cursor se encuentra en modo **K** y pulsamos una tecla alfabética, la palabra clave correspondiente aparece en la pantalla. Ahora bien, además de esto, suceden dos cosas más: el cursor se desplaza tras la palabra clave, para indicar que la siguiente escritura se realizará a continuación y, sobre todo, el cursor abandona su forma de **K** y se convierte en una **L**.

Esta **L** es la inicial de la palabra inglesa *LITERAL* (al igual que **K** lo es de *Keyword)* y con ello, el Spectrum nos anuncia que las teclas que sean pulsadas a continuación, serán interpretadas «literalmente», es decir, tal como son. Por tanto, en el modo **L**, los números y las letras se obtendrán pulsando simplemente las teclas correspondientes.

En este modo, el ordenador se comporta igual que si fuera una máquina de escribir, en la que las mayúsculas se obtienen pulsando simultáneamente la tecla de **CAPS SHIFT** (mayúsculas) y la de la letra que queremos escribir. Así pues, en el modo **L**, si pulsamos la tecla **B** aparecerá la letra **b**, y si además mantenemos pulsada a un tiempo la tecla **CAPS SHIFT** aparecerá la **B**. Como es lógico, las mayúsculas de los números no existen y por tanto, al pulsar un número al tiempo que se tiene presionada la tecla de **CAPS SHIFT**, no aparecerá ningún número, sino que se producirá un efecto diferente que a continuación estudiaremos.



TECLAS NUMERICAS TECLAS ALFANUMERICAS TECLAS DE CONTROL

LAS TECLAS DE CONTROL

Hasta ahora hemos visto como se comportan las teclas alfabéticas y las numéricas cuando se pulsan en los modos **K** y **L**. Utilizar estas teclas siempre supone escribir algo en la pantalla, ya sean letras, números o palabras clave. Ahora bien, en el teclado del Spectrum existen otras teclas diferentes, que no son números ni letras y que no tienen la misión de escribir nada en la pantalla, sino realizar determinadas funciones que más adelante veremos. Estas teclas diferentes, se denominan TECLAS DE CONTROL.

Empezaremos por una que ya hemos utilizado anteriormente: **CAPS SHIFT.** Como ya sabemos, una de sus misiones es la de obtener las mayúsculas de las letras cuando se pulsa al mismo tiempo que éstas. Si probamos a pulsar la tecla de **CAPS SHIFT** y ninguna otra al tiempo, veremos que no ocurre nada en la pantalla. Esto se debe a que es una tecla de control y, como ya hemos dicho, no tiene la misión de escribir nada nuevo en la pantalla, sino sólo de controlar el paso a mayúsculas.

En el teclado hay otro tipo de SHIFT muy importante: la tecla de **SYMBOL SHIFT.** Su misión es doble. Por una parte, nos sirve para pasar el cursor de modo **K** a modo **E** y viceversa, lo cual se produce cuando es utilizada conjuntamente con la tecla **CAPS SHIFT.** Por tratarse de dos teclas de control **(CAPS SHIFT** y **SYMBOL SHIFT),** no escriben nada nuevo en la pantalla, sino que sólo cambian el tipo de cursor. La segunda función de **SYMBOL SHIFT** la veremos a continuación.

Las teclas alfabéticas del ordenador tienen escrito sobre ellas tres tipos diferentes de datos: la letra a que corresponden, su palabra clave y un símbolo o palabra más, que en el Spectrum estándar aparece en color rojo (como la tecla de **SYMBOL SHIFT),** y en el modelo Plus entre la palabra clave y la letra correspondiente.

La pulsación de una tecla alfabética al tiempo que la de **SYMBOL SHIFT,** hará aparecer en la pantalla ese tercer dato de la tecla (palabra o símbolo). Esto ocurrirá en todos los modos del cursor, excepto en el **E** y en el **G,** es decir, en los modos **K, L** y **C.**

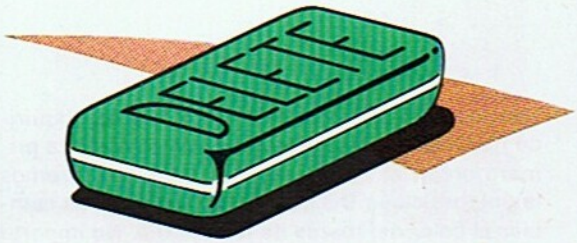
Si en vez de una tecla alfabética, se pulsase una numérica (además de **SYMBOL SHIFT),** aparecerían los caracteres que se señalan en rojo sobre las teclas numéricas del Spectrum estándar o las que están situadas en la parte superior derecha de las teclas numéricas del Plus, que lógicamente son los mismos: **(!, @, #, $, %, &, ', (,)** y **_).**

Aunque más adelante volveremos sobre el tema de las teclas de control, vamos a hablar aún de dos más de ellas, que nos permitirán comenzar a darle algunas sencillas órdenes a nuestro Spectrum.

La primera de ellas es la función de borrado (en inglés ***DELETE).*** Esta se obtiene en el Spectrum Plus mediante una tecla situada al comienzo de la segunda fila del teclado; en el modelo normal, mediante la pulsación de **CAPS SHIFT** y **0** a un mismo tiempo.

Respecto a las diferencias de teclas entre el Spectrum estándar y el modelo Plus, hemos de

La tecla de CONTROL DELETE, actúa como una goma de borrar en la pantalla de nuestro televisor.



II. Los códigos de color en el Spectrum son:

Black	Negro	0
Blue	Azul	1
Red	Rojo	2
Magenta	Magenta	3
Green	Verde	4
Cyan	Cián	5
Yellow	Amarillo	6
White	Blanco	7

ir aclarando que cualquier sistema de escritura que especifiquemos para el modelo normal servirá para el Plus, aunque éste disponga de alguna tecla especial a tal efecto. Es decir, si la función de borrado **(DELETE)**, se consigue en el modelo estándar mediante la acción simultánea sobre las teclas **CAPS SHIFT** y **0**, el mismo efecto conseguiremos en el Plus, aunque en éste hubiera sido más cómodo pulsar la tecla que incorpora esa función específica (en nuestro caso la de **DELETE**).

DELETE nos permitirá borrar lo último que escribimos, aunque de momento, en caso de apuro, el sistema más eficaz de borrado es sin duda desconectar y volver a conectar el ordenador.

La última tecla de control de la que hablaremos en este capítulo, es seguramente la más importante de todas: **ENTER**. Hasta el momento, hemos aprendido como escribir cosas en la pantalla, pero no hemos visto que nuestro Spectrum hiciera nada más con respecto a lo que hemos escrito. Esto se debe a que el ordenador nos permite escribir y borrar todas las veces que queramos, hasta estar seguros de que la orden escrita es la que queremos realmente darle.

Una vez que estemos seguros de la corrección de la instrucción, pulsemos **ENTER** y el ordenador realizará a partir de ese momento dos tareas: en primer lugar, comprobar que ha comprendido la orden que le hemos dado y, una vez hecho ésto, cumplirla.

Por tanto, utilizar la tecla **ENTER** es algo así como decirle al Spectrum: «Ya he terminado de preparar la orden. Entérate de lo que quiero y hazlo».

NUESTRA PRIMERA ORDEN

Con todos los conocimientos que hemos adquirido hasta el momento, vamos a darle nuestra primera orden al Spectrum. Para ello, utilizaremos la palabra clave **BORDER**, cuya misión es cambiar el color del marco de la pantalla. No importa que no dispongamos de una T.V. en color, porque también podremos apreciar las tonalidades diferentes en una en blanco y negro.

La instrucción **BORDER** ha de seguirse de un número, que indica el color del que deseamos «pintar» el marco de la pantalla. Estos códigos aparecen en la tabla adjunta, y van del 0 (negro) al 7 (blanco). Como recordatorio, los nombres ingleses de los colores están escritos en el teclado del Spectrum, encima de la tecla numérica que les corresponde.

La tecla de CONTROL ENTER, es la entrada que el ordenador utiliza para dejar pasar nuestras órdenes.

Adelante. Probemos los diferentes colores. Como pista ahí va la forma de obtener el primero, el resto es tarea nuestra conseguirlos. Así pues, finalizaremos describiendo los pasos necesarios para escribir **BORDER 0**.

Para obtener el marco negro (código de color 0), hay que situar el cursor en modo **K**, puesto que lo primero que hemos de escribir es **BORDER**, que es una palabra clave. Una vez hecho ésto, presionamos la tecla **B**, con lo que obtendremos la palabra clave **BORDER**.

El cursor pasará a **L**, indicando que ya no se esperan más palabras clave. Efectivamente, ahora está esperando un número que le indique el color del que queremos el marco, en el ejemplo el 0, por ser el código de negro (este número es el que deberemos ir cambiando para obtener el resto de los colores). Ya sólo nos queda confirmarle nuestra orden, pulsando la tecla **ENTER**.

¡!

El CURSOR tiene dos misiones fundamentales:
1. Indicar cual es el lugar en que se comenzará a escribir el próximo dato tomado del teclado.
2. Informar al programador sobre el tipo de dato que el ordenador está esperando.

Cuando el cursor se encuentra en el modo **K**, las teclas pulsadas serán interpretadas como palabras clave. Si estuviera en modo **L**, las teclas serían interpretadas como en una máquina de escribir.

*****

La misión de la tecla **CAPS SHIFT** es doble:
1. Cuando el cursor se encuentra en modo **L**, sirve para pasar a escribir en mayúsculas las letras correspondientes a las teclas pulsadas.
2. Combinada con las teclas numéricas (fila superior), cualquiera que sea el modo en que se encuentre el cursor, realizará las funciones de las teclas de control.

JOYSTICKS

DADME un *joystick* y ganaré la carrera». Bueno, no se sabe a ciencia cierta si eso era lo que pensaba Angel Nieto en la parrilla de salida del último Gran Premio. Pero si nuestra «máquina» es un Spectrum, y los mandos sus teclas, quizás encontremos más dificultades de las inicialmente previstas para convertirnos en campeones del mundo, batir el récord de los 100 metros lisos, o realizar una peligrosa aproximación aérea a la base «Zulú-Tango Rojo».

Sufridos todos estos problemas, pronto pensaremos en acoplar a nuestro Spectrum algún mando especial para el manejo de juegos. Los hay de los más diversos tipos. Unos tienen forma de bola *(trackball)*, otros funcionan por control remoto *(wireless)*, pero los más comunes son los de palanca *(joystick)*.

La utilidad de un *joystick* a la hora de manejar un juego es indudable; ahora bien, ¿cómo funcionan estos curiosos artefactos?

En el mercado existen un gran número de joysticks e interfaces destinados al control de juegos de acción.

EL INTERIOR DEL *JOYSTICK*

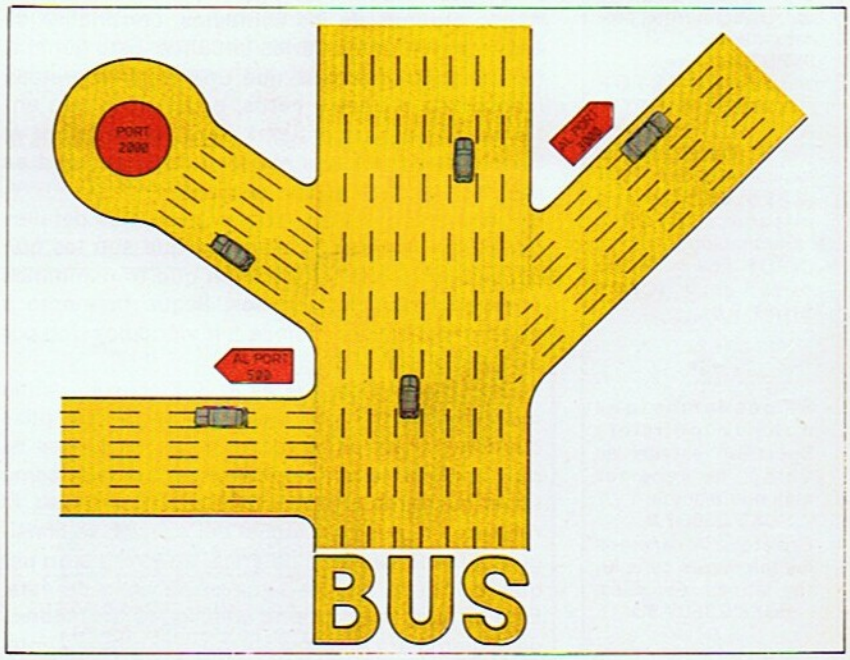
Este periférico está formado por dos partes principales: la palanca *(stick)* y la base o caja.

Mediante el movimiento de la palanca, ya sea en cruz o en diagonal, podremos comunicar al ordenador los deplazamientos que deseamos realizar por la pantalla.

La palanca suele incorporar unas hendiduras en donde se acoplan los dedos del jugador. Además, la mayoría de los *joysticks* de diseño moderno, poseen un pulsador, ya sea en la parte superior de la palanca para ser utilizado con el pulgar, o bien en forma de gatillo, que puede ser manejado con el índice; o incluso ambas formas de pulsador, cuya misión es servir de botón de disparo en los juegos de acción.

Bajo la palanca está la caja, que sirve de apoyo al *stick*. Generalmente, en la parte inferior encontramos unas ventosas, con el fin de fijar el conjunto caja-palanca a la mesa. Algunos modelos tienen, además, uno o más botones de disparo situados en la base.

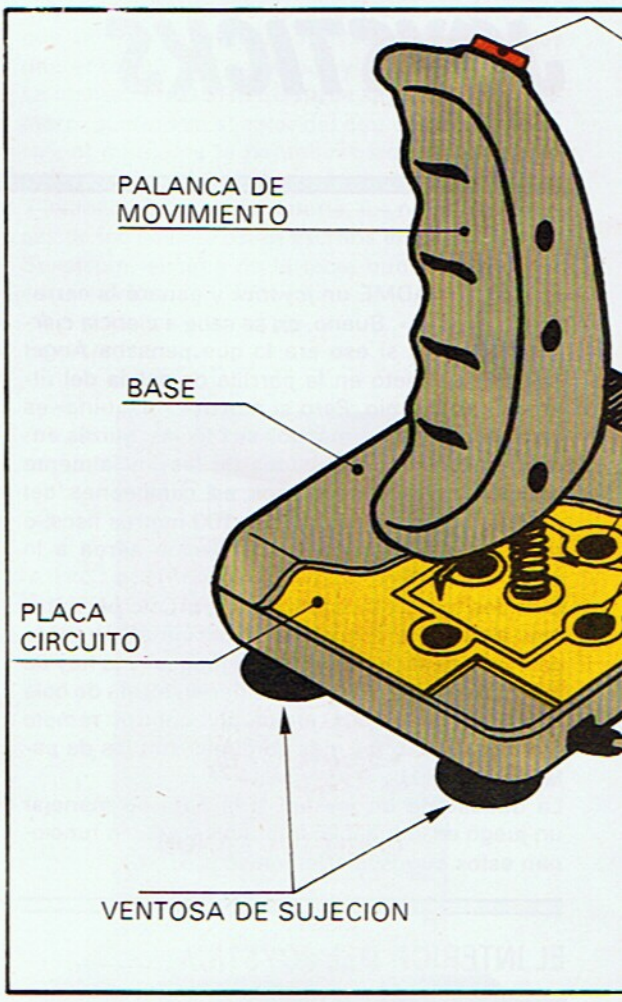
El BUS es el conjunto de circuitos por el que circulan los datos, hasta llegar a algún PORT.



Finalmente, los modelos más sofisticados, poseen también en su base un interruptor deslizante, conocido generalmente como *auto-fire* (disparo automático). En su posición de desconexión *(off)*, no realiza misión alguna, pero cuando se encuentra conectado *(on)*, produce el mismo efecto que si estuviéramos pulsando constantemente el botón de disparo.

Desde la caja, parte un cable (1 metro o metro y medio es su longitud habitual) que acaba en un conector del tipo **CANON** o **D**, con 9 terminales circulares dispuestos en dos filas (cinco y cuatro salidas, respectivamente). A través de él, circulan las señales eléctricas necesarias para la comunicación entre el *joystick* y el ordenador.

Alojados en el interior de la base, sobre un circuito impreso, se encuentran distribuidos en forma de cruz cuatro pequeños interruptores o contactos, orientados en las direcciones vertical y horizontal. Concretamente, uno arriba, otro abajo, el tercero a la izquierda y el cuarto a la derecha. En la parte superior de la palanca, suele encontrarse un quinto interruptor, correspondiente al botón de disparo. Los *joysticks* con pulsadores en la base, disponen también de contactos sobre la placa del circuito impreso.



El **joystick** *es, sin duda, el mando de juego que goza de mayor aceptación.*

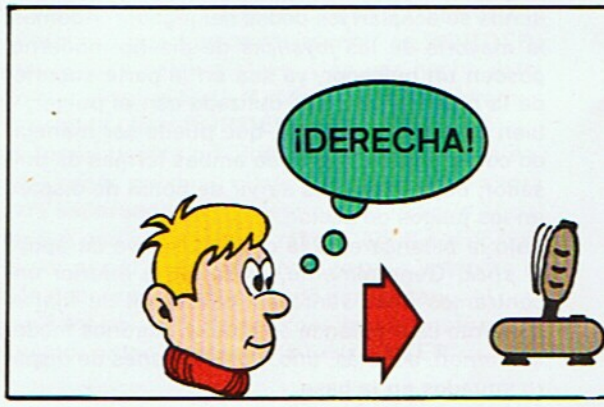
FUNCIONAMIENTO DEL *JOYSTICK*

Cuando movemos el *stick* en cualquier sentido, o pulsamos el botón de disparo, estamos presionando algunos de los contactos, cerrándose de esta manera uno o varios circuitos. Esto generará una señal eléctrica, que una vez interpretada (convertida a unos y ceros; es lo único que entiende nuestro ordenador), será recogida por el Spectrum y hará que el móvil de la pantalla se desplace en la dirección deseada.

En todo este proceso, existen pequeños detalles de los que conviene hablar, ya que son los que consiguen que la información que transmitimos al *joystick* cuando jugamos, llegue fielmente a nuestro ordenador. Vamos a ir viéndolos uno por uno.

Hemos hablado de qué es lo que ocurre cuando movemos la palanca, pero no de lo que pasa cuando no queremos moverla. En los juegos es muy importante tanto moverse con rapidez, como saber quedarse quieto en el momento preciso. El retorno a la posición inicial del *joystick* se consigue dejando suelta la palanca, en virtud a un pequeño muelle que se sitúa en la base de ésta. Esta es el mismo sistema que siguen los botones de disparo para no estar pulsados continuamente.

Otro punto a destacar es cómo se consiguen los movimientos diagonales. Cada vez que realizamos uno de los desplazamientos fundamentales, oprimimos el correspondiente contacto en la placa. Ahora bien, cuando el movimiento es en dia-



BITS

He aquí el sistema para escribir en la última línea de la pantalla, reservada inicialmente a los mensajes del Spectrum.

```
10 PRINT#0;"MENSAJE"
20 GOTO 20
```

*

Abandonar un **INPUT** durante la ejecución de un programa, puede convertirse en un problema, ya que se deshabilita la función **BREAK**. Según el tipo de **INPUT**, optemos por los siguientes procedimientos:

INPUT numérico = Introducir **STOP** (**SYMBOL SHIFT** + **A**).

INPUT literal = Borrar las comillas con **EDIT** (**CAPS SHIFT** + **1**) e introducir **STOP** a continuación.

INPUT line = Pulsar cursor abajo (**CAPS SHIFT** + **6**).

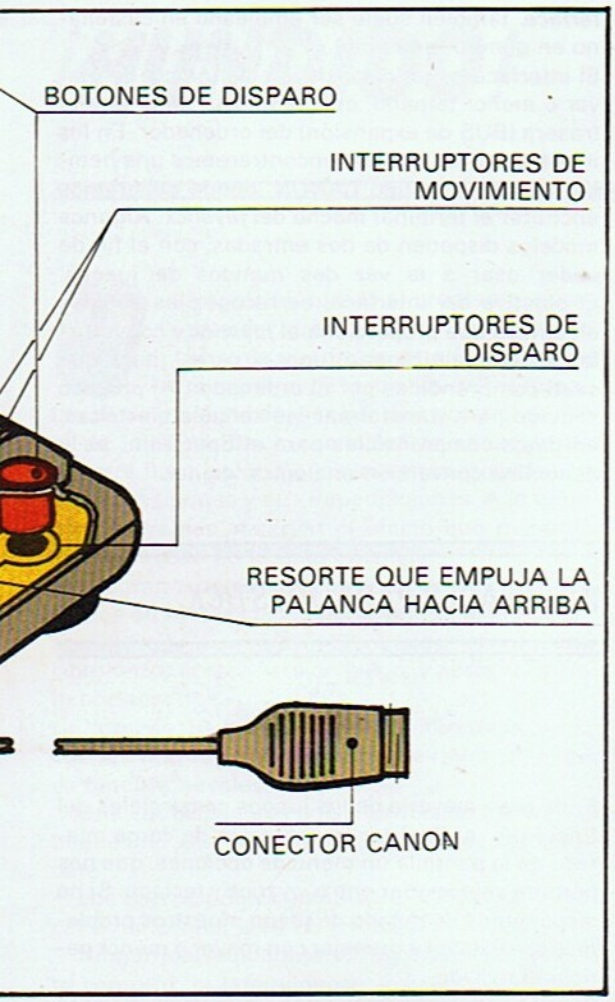
*

Si deseamos que nuestro poliglota Spectrum escriba en Chino, no tenemos más que ejecutar

POKE 23607,0

Pronto añoraremos los familiares caracteres latinos. Así pues,

POKE 23607,60



gonal, los contactos afectados son los dos que componen el sentido deseado. Por ejemplo, al realizarse un desplazamiento en la diagonal superior derecha, se realiza una presión simultánea sobre los contactos de arriba y derecha.

La mayoría de los juegos controlables por *joystick,* se basan en la rapidez de reflejos para el movimiento y para el disparo. Debido a esto último, algo muy a tener en cuenta a la hora de adquirir uno de estos periféricos, es la comodidad de la que vayamos a disfrutar cuando hagamos fuego contra una pléyade de marcianitos. En este sentido, los *joystick* incorporan sus pulsadores de disparo en lugares estratégicos, de forma que puedan ser cómodamente accionados.

La opción de *auto-fire* (disparo automático), es sin duda uno de los grandes inventos para los jugadores nerviosos, que no cesan en su afán de destruir las naves enemigas. Para evitar el uso desenfrenado del botón de disparo, algunos juegos se programan de manera que no se puede volver a hacer fuego hasta que no se libera el pulsador de disparo. En estos casos, el hecho de disparar repetidamente resulta algo cansado. Los diseñadores también han pensado en ésto, y por tanto, la opción de *auto-fire* no consiste en una acción continua sobre el pulsador de disparo, sino de forma intermitente y a gran velocidad.

Finalmente, estudiaremos el punto culminante del proceso de la transmisión de datos al ordenador: la codificación de los movimientos en forma comprensible para el Spectrum.

UN INTERMEDIARIO PARA EL ORDENADOR: EL INTERFACE

Antes de continuar, vamos a explicar dos conceptos que serán necesarios de ahora en adelante: **BUS** y **PORT.**

BUS es el conjunto de conductores eléctricos contenidos en un ordenador. A través de él, se realizan los intercambios de información entre los diferentes elementos del Spectrum. Podemos

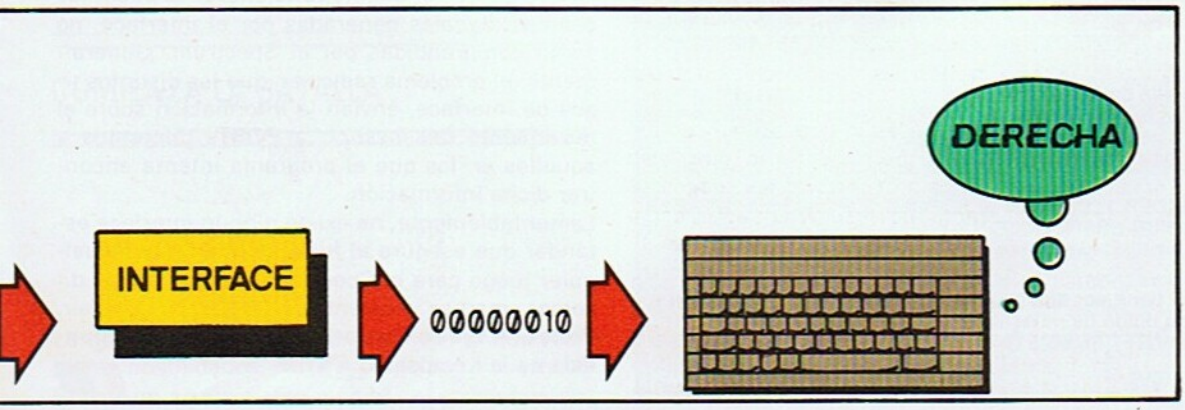
El interface para* joystick, *actúa como conversor analógico/digital de las órdenes transmitidas por el usuario del ordenador.

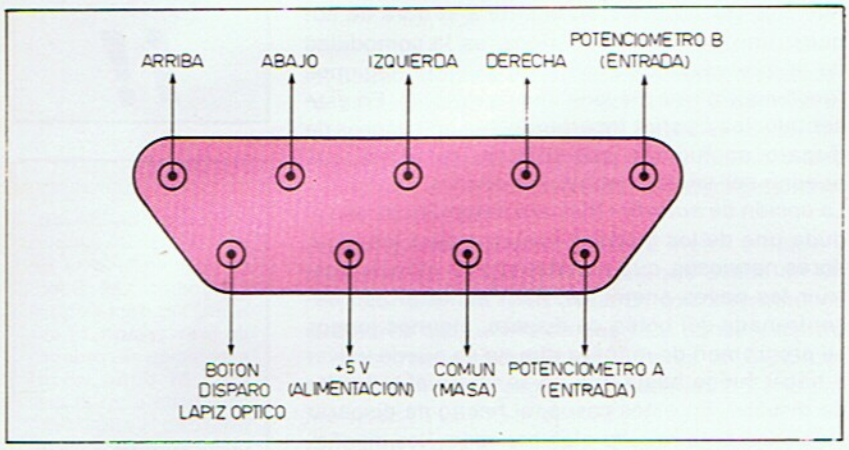
¡!

Recordemos no conectar NUNCA ningún dispositivo al BUS de expansión del Spectrum, sin haber cortado previamente la alimentación del ordenador. El daño en el Spectrum o en el dispositivo conectado, puede ser irreparable.

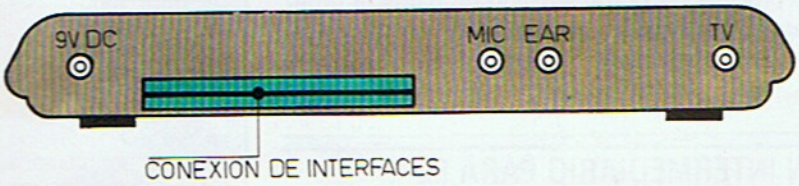
*

Un gran número de pulsadores en un *joystick,* no supone que éste sea más completo, sino simplemente más cómodo de manejar, ya que todos los botones de disparo realizarán exactamente la misma función.





El conector CANON, supone una norma prácticamente estandarizada en los joysticks.



La conexión de los interfaces al Spectrum, se realiza a través de su BUS de expansión.

pensar en el BUS como en una autopista de varios carriles, en la que los coches que circulan por ella fueran los elementos de información. Y podemos imaginarnos un PORT como los terminales donde desemboca el BUS. Desde los PORTs, la información que les ha llegado por el BUS correspondiente, es transmitida a la memoria central o al periférico requerido.

Cuando hablamos del conector CANON, con sus nueve terminales dispuestos en forma de D, omitimos deliberadamente el lugar donde se enchufaba. Desgraciadamente, el Spectrum no es capaz de interpretar directamente las señales eléctricas que el *joystick* emite, y para ello será preciso el uso de un interface (el término inglés *interface,* también suele ser empleado en castellano en género femenino).

El interface es un dispositivo electrónico, de mayor o menor tamaño, que se encaja en la tarjeta trasera (BUS de expansión) del ordenador. En los interfaces para *joystick,* encontraremos una hembra de conector tipo CANON, donde deberemos enchufar el terminal macho del *joystick.* Algunos modelos disponen de dos entradas, con el fin de poder usar a la vez dos mandos de juegos. El objetivo del interface, es recoger las señales eléctricas que proporciona el *joystick* y convertirlas al lenguaje binario (unos y ceros), para que sean comprendidas por el ordenador. Al proceso seguido para transformar las señales eléctricas, en datos comprensibles para el Spectrum, se le denomina conversión analógica/digital.

INTERFACES PARA *JOYSTICK*

En la gran mayoría de los juegos comerciales del Spectrum, al finalizar la operación de carga aparece en la pantalla un menú de opciones, que nos permite seleccionar entre *joystick* y teclado. Si no disponemos de mando de juego, nuestros problemas se reducen a manejar con mayor o menor pericia el teclado.

Si por el contrario poseemos un *joystick,* a veces comprobamos con sorpresa y una cierta dosis de mal humor, que a pesar de escoger las opciones para mando de juegos, el programa no responde a los movimientos deseados.

Este tipo de problemas no son debidos al *joystick* propiamente dicho, ya que el sistema seguido por estos periféricos está muy estandarizado (salvo en modelos especiales). El problema viene, sin duda, del interface que se esté utilizando. Es absolutamente necesario que el juego a controlar esté adaptado al interface que poseamos, ya que de no ser así, no funcionará. Es decir, las señales digitales generadas por el interface, no serán comprendidas por el Spectrum. Generalmente, el problema radica en que los distintos tipos de interface, envían la información sobre el movimiento del *joystick,* a PORTs diferentes a aquellos en los que el programa intenta encontrar dicha información.

Lamentablemente, no existe ningún interface estándar que asegure el funcionamiento con cualquier juego para el Spectrum, aunque de las diversas marcas existentes (Kempston, Sinclair, Protek, A.G.F. o Cursor), sin duda la más difundida es la Kempston.

BITS

El color del borde de la pantalla de nuestro Spectrum no tiene por qué ser tan estático como el manual del equipo da a entender. He aquí dos claros ejemplos de ello:

```
10 FOR I=0 TO 255: OUT 254,I: NEXT I

10 IF INKEY$="" THEN BORDER 1: BORDER 6: BORDER 2: BORDER 5: BORDER
   2: BORDER 6: BORDER 1: PAUSE 1: GOTO 10
```

La ejecución de gráficos, no tiene por qué conseguirse a fuerza de gran número de instrucciones. Sirva como botón de muestra el siguiente ejemplo:

```
10 PLOT 140,20: DRAW OVER 1;55,55,51↑4: PLOT 130,65: DRAW 20,20,51↑4
```

¡HUNDIDO!

UNA vez ejecutado el programa, el ordenador presentará la pantalla que sirve de puente de mando. En la parte superior de la misma, se representan dos cuadrados de diez por diez posiciones. En el de la izquierda, se señalará la situación de nuestra flota, así como los impactos recibidos en nuestros buques y sus inmediaciones. A la derecha, podremos observar el efecto que nuestras salvas causan en la flota enemiga.

El programa comienza con la disposición de los barcos en el «teatro de operaciones». Esta acción, será llevada a cabo en primer lugar por nuestro cibernético compañero de juego, y posteriormente por nosotros.

Un total de 10 navíos se distribuirán de acuerdo con las normas tradicionales que rigen este tipo de batallas navales:

* Una vez dispuestos en su formación inicial, los buques no alterarán su posición durante el combate.
* Los barcos podrán situarse en dirección vertical u horizontal, pero no en diagonal.
* Ningún navío podrá tener un punto de contacto con otro; ni lateralmente, ni por sus vértices.

Gracias a las coordenadas que se señalan en el mapa correspondiente, la colocación de nuestra flotilla será muy fácil de llevar a cabo. El programa, tras notificar cuál es la longitud del buque a situar, requerirá su orientación, a lo que responderemos con **H** o **V**, según ésta sea horizontal o vertical, respectivamente. A continuación se pide la coordenada de la cabecera del barco, para lo cual se indicará en primer lugar la letra que señala su fila y, acto seguido, sin ninguna separación intermedia, el número de columna.

COMIENZA LA BATALLA

La decisión de quién será el almirante que rompa las hostilidades, corre a cargo de la suerte. El Spectrum arrojará una moneda al aire y nos notificará quién comienza el juego. A partir de ahí, y alternativamente, los jugadores podremos realizar un disparo, salvo en caso de acertar a algún buque, ya sea tocándolo o hundiéndolo, lo cual nos permitirá continuar disparando.

Las posiciones de los disparos las indicaremos mediante la introducción de las coordenadas del impacto, de forma similar a como se hace a la hora de distribuir la flota. Inmediatamente, el ordenador informará sobre el resultado de la andanada.

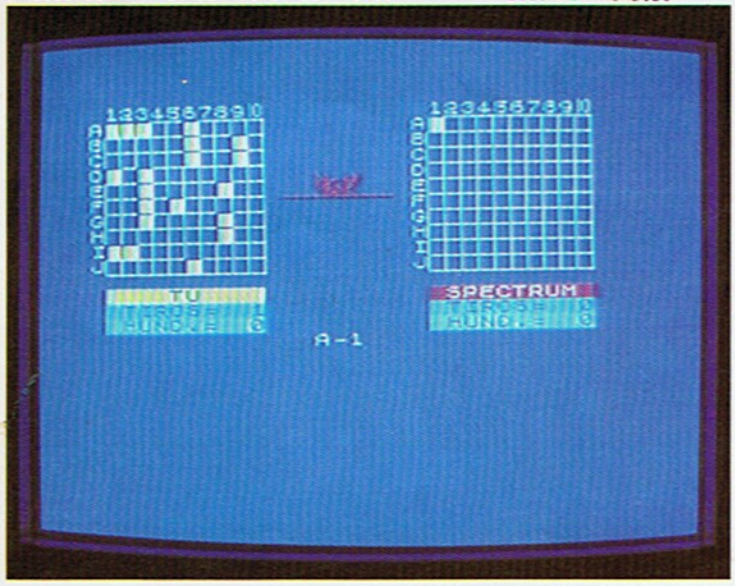
El mismo sistema seguirá el Spectrum para anunciarnos su disparo. Por supuesto, nosotros debemos indicarle los daños causados en nuestra escuadra, mediante los códigos **A**, **T** y **H**, para agua, tocado y hundido, respectivamente.

Sólo resta añadir que cuando introduzcamos el programa, hemos de tener en cuenta que las letras que aparecen subrayadas en el listado, representan los gráficos definidos de las teclas correspondientes. Una vez tecleado el programa podremos conservarlo en cinta mediante
SAVE "HUNDIDO" LINE 1

```
 10 REM ************************
 15 REM ****  J.M.MAYORAL  ****
 20 REM ************************
 25 POKE 23658,8
 30 DIM C(12,12): DIM S(12,12)
 35 GO SUB 590
 40 GO TO 1800
 45 BORDER 1: PAPER 1: INK 9
 50 POKE 23609,100: CLS
 55 REM  GRAFICO NUM.10
 60 REM
 65 RESTORE 75
 70 LET A$="ABCDEFGHIJ"
 75 DATA 0,38,41,41,41,41,38,0
 80 FOR N=0 TO 7
 85 READ F: POKE USR "P"+N,F
 90 NEXT N
 95 REM  DIBUJO CUADRICULAS
100 REM
105 FOR N=1 TO 11
110 PLOT 8,175-(8*N): DRAW 80,0
115 PLOT 168,175-(8*N): DRAW 80,0
120 PLOT 8*N,167: DRAW 0,-80
125 PLOT 160+(8*N),167: DRAW 0,-80
130 NEXT N
135 REM POSIC.LETRAS/NUMEROS
140 REM
145 FOR N=1 TO 9
150 PRINT  OVER 1;AT 0,N;N;AT 0,20+N;N;AT
N,0;A$(N);AT N,20;A$(N)
155 NEXT N
160 PRINT  OVER 1;AT 0,10;"P";AT 0,30;"P";
AT 10,0;"J";AT 10,20;"J"
165 RETURN
170 REM
175 REM  SBR. SUPERVISOR
180 REM
185 IF VH THEN  GO TO 230
190 FOR F=Y TO Y+BARCO-1
195 IF SPW=1 THEN  GO TO 220
200 IF VAL B$(2 TO )>(11-BARCO) THEN  LET
E$="IMPOSIBLE:ME SALGO DEL CUADRO": LET C1=
0: GO SUB 370: PAUSE 50: GO SUB 405: GO TO
270
205 IF C(X,F)>=1 THEN  GO TO 270
210 NEXT F
215 LET V=1: RETURN
220 IF S(X,F)>=1 THEN  LET V=0: RETURN
225 GO TO 210
230 FOR F=X TO X+BARCO-1
235 IF SPW=1 THEN  GO TO 260
240 IF N>(11-BARCO) THEN  LET E$="IMPOSIBL
E:ME SALGO DEL CUADRO": LET C1=0: GO SUB 37
0: PAUSE 50: GO SUB 405: GO TO 270
245 IF C(F,Y)>=1 THEN  GO TO 270
250 NEXT F
255 LET V=1: RETURN
260 IF S(F,Y)>=1 THEN  LET V=0: RETURN
265 GO TO 250
270 LET E$="NO HAGAS TRAMPAS"
275 LET PAP=2: LET FL=1
280 LET F1=19: LET C1=7: LET V=0
285 GO SUB 360: BEEP 1.5,-10: GO SUB 395:
RETURN
290 REM  SBR. RELLENO
295 REM
300 LET COMP=((BARCO-1)*VH)
305 FOR F=X-1 TO (X+1)+COMP
310 FOR C=Y-1 TO (Y+BARCO)-COMP
315 IF SPW=1 THEN  GO TO 345
320 IF C(F,C)>1 THEN  GO TO 345
325 LET C(F,C)=1
330 NEXT C
335 NEXT F
340 RETURN
345 IF S(F,C)>1 THEN  GO TO 330
350 LET S(F,C)=1
355 GO TO 330
360 REM  SBR. ESCRIT.HORIZ.
365 REM
370 FOR N=1 TO LEN E$
375 PRINT  PAPER PAP; INK 9; FLASH FL;AT F
1,N+C1;E$(N)
380 BEEP .01,20
385 NEXT N
390 RETURN
395 REM  SBR. BORRADO E.H.
400 REM
405 FOR N=1 TO LEN E$
410 PRINT  PAPER 1;AT F1,C1+N;" "
415 NEXT N
420 RETURN
425 REM  SBR.VERT/HOR. P.ORIG
430 REM
435 LET SPW=1
440 LET VH=INT (RND*2)
445 IF VH=1 THEN  GO TO 465
450 LET X=INT (RND*10)+2
455 LET Y=INT (RND*(11-BARCO))+2
460 RETURN
465 LET Y=INT (RND*10)+2
470 LET X=INT (RND*(11-BARCO))+2
475 RETURN
480 REM  SBR. STRIN>>>NUMEROS
485 REM
490 FOR N=1 TO LEN A$
495 IF B$(1)=A$(N) THEN  LET X=N+1: LET Y=
1+VAL B$(2 TO ): RETURN
500 NEXT N
505 IF B$="" OR CODE B$(1)<65 OR CODE B$(1
)>74 OR VAL B$(2 TO )<1 OR VAL B$(2 TO )>10
THEN  GO TO 515
510 LET V=1: RETURN
515 LET E$=" TIRO INCORRECTO "
520 LET FL=1: LET F1=18: LET C1=7: LET PAP
=2
525 GO SUB 370: PAUSE 40: GO SUB 405
530 LET V=0: RETURN
535 PRINT AT 19,0;X,Y
540 IF O(X-1,Y-1) THEN  GO TO 570
545 REM TIRO NO REPETIDO
550 LET O(X-1,Y-1)=1
555 PRINT AT 15,13;" ";B$(1);"-";B$(2 TO )
560 BEEP .3,15: PAUSE 10: LET V=1: RETURN
565 REM TIRO REPETIDO
570 LET E$=" TIRO REPETIDO "
575 LET F1=18: LET C1=7: LET FL=1: LET PAP
=6
580 GO SUB 370: PAUSE 60: GO SUB 405
585 LET V=0: RETURN
590 LET A$="abcdefghijkl"
595 RESTORE 600
600 DATA 32,50,57,60,63,59,63,62
605 DATA 0,0,0,160,225,195,138,221
610 DATA 16,56,124,252,248,248,240,240
615 DATA 31,15,7,3,1,0,0,0
620 DATA 222,175,246,233,253,250,127,62
625 DATA 224,64,224,176,24,60,242,255
630 DATA 0,0,0,0,32,50,57,60
635 DATA 0,0,0,0,0,0,0,160
640 DATA 0,0,0,0,16,56,124,252
645 DATA 63,59,63,62,31,15,7,3
650 DATA 225,195,138,221,222,175,246,233
655 DATA 248,248,240,240,224,64,224,176
660 FOR K=1 TO LEN A$
665 FOR N=0 TO 7
670 READ D
675 POKE USR A$(K)+N,D
680 NEXT N
685 NEXT K
690 RETURN
695 REM  ANIMACION
700 PLOT 96,127: DRAW 54,0
705 PAUSE 10
710 LET B$="ABCDEFGHIJKL"
715 PRINT AT 4,14;B$( TO 3)
720 PRINT AT 5,14;B$(4 TO 6)
725 PAUSE 50
730 PRINT AT 4,14;B$(7 TO 9)
735 PRINT AT 5,14;B$(10 TO 12)
740 PAUSE 10
745 PRINT AT 4,14;"   "
750 PRINT AT 5,14;"ABC"
755 PAUSE 10
760 PRINT AT 5,14;"GHI"
765 PAUSE 10
770 PRINT AT 5,14;"   "
775 RETURN
780 REM COMPROBADOR SP
785 LET BARCO=C(XT,YT)-1
790 IF C$="A" AND C(XT,YT)<=1 THEN  GO TO
845
795 IF C$="T" AND C(XT,YT)>1 AND BARCO-BCT
-1<>0 THEN  LET BCT=BCT+1: GO TO 845
800 IF C$="H" AND C(XT,YT)>1 AND BARCO-BCT
-1=0 THEN  LET BU=1: GO TO 845
805 LET E$="TRAMPOSO:NO ACEPTO TU RESPUEST
A!"
810 LET C1=-1: LET F1=21: LET V=1
815 LET PAP=2: LET FL=1: GO SUB 370
820 FOR N=1 TO 5
825 BEEP .1,10: BEEP .2,5
830 NEXT N
835 GO SUB 405
840 RETURN
845 LET V=0: RETURN
850 REM SP ESPERA RESPUESTA
855 PRINT AT 21,2; PAPER 7; INK 1; FLASH 1
;" ESPERO RESPUESTA  (A/T/H) "
860 IF INKEY$="" THEN  GO TO 860
865 LET C$=INKEY$
870 BEEP .2,35
875 PRINT  PAPER 1;AT 21,0;"
                            "
880 RETURN
885 REM DISPL. DISPARO/CONTSP
890 IF BT=1 THEN  GO TO 900
895 LET XT=X: LET YT=Y
900 LET T$=A$(XT-1)
905 PRINT AT 18,14; PAPER 7;T$;"-";YT-1
910 BEEP .1,45
915 REM CONTADOR & DISPLAY
920 LET CONTSP=CONTSP+1
925 PRINT AT 13,31-LEN STR$ CONTSP; PAPER
5; INK 0;CONTSP
930 RETURN
935 REM MENSAJE TIRO SP
940 PRINT  PAPER 1;AT 15,13;"    ";AT 17,1
1;"          "
945 LET E$=" TIRO YO! "
950 PRINT  PAPER 1;AT 18,14;"     "
955 LET PAP=7: LET F1=16: LET C1=10
960 LET FL=0
965 GO SUB 370: RETURN
970 REM DISPARADO & COMPROBADOR
    DE TIRO
975 IF BT=1 THEN  GO TO 1035
980 IF CONTSP>=30 THEN  GO TO 1010
985 LET X=INT (RND*10)+2
990 LET Y=INT (RND*10)+2
995 IF T(X,Y)=-1 THEN  LET V=1: GO TO 980
1000 LET T(X,Y)=-1
1005 RETURN
1010 LET X=INT (RND*10)+2
1015 FOR F=2 TO 11
1020 IF T(X,F)=0 THEN  GO TO 1030
1025 NEXT F: GO TO 1010
1030 LET Y=F: RETURN
1035 IF T(XT,YT)=-1 THEN  LET V=0: RETURN
1040 LET T(XT,YT)=-1: LET V=1: RETURN
1045 REM DISPLAYA EFECT. TOCADO
1050 PRINT  PAPER 2; OVER 1; BRIGHT 1;AT XT
-1,YT-1;" "
1055 REM BORRO COORD. TIRO SP
1060 PRINT  PAPER 1;AT 18,24;"    "
1065 RETURN
1070 REM RELLENO A  -1
1075 LET COMP=((BARCO-1)*VH)
1080 FOR F=X-1 TO X+1+COMP
1085 FOR C=Y-1 TO Y+BARCO-COMP
1090 LET T(F,C)=-1
1095 NEXT C
1100 NEXT F
1105 RETURN
1110 REM IMPRIME AGUA: *
1115 PRINT  OVER 1; INK 7;AT XT-1,YT-1;"*"
1120 RETURN
1125 REM SEGUIDOR TOCADOS SP
1130 LET X1=X: LET X2=X
1135 LET X3=X: LET X4=X
1140 LET Y1=Y: LET Y2=Y
1145 LET Y3=Y: LET Y4=Y
1150 LET D=INT (RND*4)+1
1155 RANDOMIZE : IF D(D)=1 THEN  GO TO 1150
1160 GO TO 1050+(D*150)
1165 REM BORRA MENSAJE
1170 LET E$="                    "
1175 LET C1=10: LET F1=16
1180 GO SUB 405
1185 PRINT  PAPER 1;AT 18,14;"    "
1190 RETURN
1195 REM TIRO HORIZ./DECHA SP
1200 IF D(1)=1 THEN  GO TO 1150
1205 IF Y1+1=12 THEN  LET D(1)=1: GO TO 115
0
1210 LET Y1=Y1+1
1215 LET XT=X1
1220 LET YT=Y1
1225 GO SUB 975
1230 IF V=0 THEN  LET D(1)=1: GO TO 1150
1235 GO SUB 940
1240 GO SUB 890
1245 GO SUB 855
1250 GO SUB 780
1255 IF V=1 THEN  GO TO 1245
1260 GO SUB 1170
1265 IF C$="A" THEN  GO TO 1295
1270 IF BU=1 THEN  GO TO 1315
1275 LET D(3)=1: LET D(4)=1
1280 GO SUB 1050
1285 LET X1=XT: LET Y1=YT
1290 GO TO 1205
1295 LET SPW=0
1300 GO SUB 1115
1305 LET XU=X: LET YU=Y
1310 RETURN
1315 LET VH=0: LET Y=YT-(BARCO-1)
1320 GO SUB 1050
1335 LET X=XT
1340 RETURN
1345 REM TIRO HORIZ./IZQDA SP
1350 IF D(2)=1 THEN  GO TO 1150
1355 IF Y2-1=1 THEN  LET D(2)=1: GO TO 1150
1360 LET Y2=Y2-1
1365 LET XT=X2
1370 LET YT=Y2
1375 GO SUB 975
1380 IF V=0 THEN  LET D(2)=1: GO TO 1150
1385 GO SUB 940
1390 GO SUB 890
1395 GO SUB 855
1400 GO SUB 780
```

```
1405 IF V=1 THEN  GO TO 1395
1410 GO SUB 1170
1415 IF C$='A' THEN  GO TO 1445
1420 IF BU=1 THEN  GO TO 1465
1425 LET D(3)=1: LET D(4)=1
1430 GO SUB 1050
1435 LET X2=XT: LET Y2=YT
1440 GO TO 1355
1445 LET SPW=0
1450 GO SUB 1115
1455 LET XU=X: LET YU=Y
1460 RETURN
1465 LET VH=0: LET Y=YT
1470 GO SUB 1050
1485 LET X=XT
1490 RETURN
1495 REM TIRO VERT./ABAJO SP
1500 IF D(3)=1 THEN  GO TO 1150
1505 IF X3+1=12 THEN  LET D(3)=1: GO TO 115
0
1510 LET X3=X3+1
1515 LET XT=X3
1520 LET YT=Y3
1525 GO SUB 975
1530 IF V=0 THEN  LET D(3)=1: GO TO 1150
1535 GO SUB 940
1540 GO SUB 890
1545 GO SUB 855
1550 GO SUB 780
1555 IF V=1 THEN  GO TO 1545
1560 GO SUB 1170
1565 IF C$='A' THEN  GO TO 1595
1570 IF BU=1 THEN  GO TO 1615
1575 LET D(1)=1: LET D(2)=1
1580 GO SUB 1050
1585 LET X3=XT: LET Y3=YT
1590 GO TO 1505
1595 LET SPW=0
1600 GO SUB 1115
1605 LET XU=X: LET YU=Y
1610 RETURN
1615 LET VH=1: LET X=XT-(BARCO-1)
1620 GO SUB 1050
1635 LET Y=YT
1640 RETURN
1645 REM TIRO VERT./ARRIBA SP
1650 IF D(4)=1 THEN  GO TO 1150
1655 IF X4-1=1 THEN  LET D(4)=1: GO TO 1150
1660 LET X4=X4-1
1665 LET XT=X4
1670 LET YT=Y4
1675 GO SUB 975
1680 IF V=0 THEN  LET D(4)=1: GO TO 1150
1685 GO SUB 940
1690 GO SUB 890
1695 GO SUB 855
1700 GO SUB 780
1705 IF V=1 THEN  GO TO 1695
1710 GO SUB 1170
1715 IF C$='A' THEN  GO TO 1745
1720 IF BU=1 THEN  GO TO 1765
1725 LET D(1)=1: LET D(2)=1
1730 GO SUB 1050
1735 LET X4=XT: LET Y4=YT
1740 GO TO 1655
1745 LET SPW=0
1750 GO SUB 1115
1755 LET XU=X: LET YU=Y
1760 RETURN
1765 LET VH=1: LET X=XT
1770 GO SUB 1050
1785 LET Y=YT
1790 RETURN
1795 REM
1800 REM  PROGRAMA PRINCIPAL
1805 REM
1810 GO SUB 45
1815 LET E$='  ESTOY COLOCANDO MIS BARCOS
'
1820 LET PAP=6: LET FL=1:
1825 LET F1=13: LET C1=0
1830 GO SUB 370
1835 REM ) COLOC/BARC. SPECTR.
1840 REM
1845 LET SPW=1
1850 FOR B=4 TO 1 STEP -1
1855 LET BARCO=B
1860 FOR P=1 TO (5-BARCO)
1865 GO SUB 435
1870 GO SUB 185
1875 IF V=0 THEN  GO TO 1865
1880 IF VH=1 THEN  GO TO 1920
1885 FOR F=Y TO (Y+BARCO-1)
1890 LET S(X,F)=BARCO+1
1895 NEXT F
1900 GO SUB 300
1905 NEXT P
1910 NEXT B
1915 GO TO 1940
1920 FOR F=X TO (X+BARCO-1)
1925 LET S(F,Y)=BARCO+1
1930 NEXT F
1935 GO TO 1900
1940 GO SUB 405
1945 LET E$='  TERMINE.  '
1950 LET PAP=2: LET FL=1:
1955 LET F1=14: LET C1=9
1960 GO SUB 370
1965 PAUSE 80
1970 GO SUB 405
1975 REM  COLOC.CONTRINCANTE
1980 LET E$='AHORA COLOCA LOS TUYOS'
1985 LET PAP=5
1990 LET F1=15: LET C1=4:
1995 GO SUB 370: PAUSE 70
2000 LET SPW=0
2005 FOR B=4 TO 1 STEP -1
2010 LET BARCO=B
2015 PRINT  PAPER 2; INK 6; FLASH 1;AT 13,3
;'66666666666666666666666';AT 17,3;'66666
66666666666666666666'
2020 FOR G=1 TO 3: PRINT  PAPER 2; INK 6; F
LASH 1;AT 13+G,3;'6';AT 13+G,27;'6': NEXT G
2025 FOR P=1 TO (5-BARCO)
2030 PRINT AT 15,5;'BARCO DE ';B;' POSICION
ES '
2035 IF 5-BARCO=4 THEN  GO TO 2070
```

```
2040 LET E$='   VERTICAL U HORIZONTAL ?  ':
2045 LET PAP=3: LET F1=21: LET C1=1
2050 GO SUB 370
2055 INPUT '(V/H)', LINE V$: GO SUB 405
2060 IF V$='V' OR V$='H' THEN  GO TO 2070
2065 LET E$=' ERRONEO ': LET FL=1: LET F1=2
0: LET C1=10: LET PAP=7: GO SUB 370: BEEP 1
.5,-15: GO SUB 405: GO TO 2040
2070 LET E$=' INTRODUCE COORDENADA INICIAL
'
2075 LET PAP=2: LET F1=21: LET C1=0
2080 GO SUB 370
2085 INPUT '(FILA/COLUMNA)', LINE B$: GO SU
B 405
2090 IF LEN B$<=1 THEN  GO TO 2105
2095 IF CODE B$(1)<65 OR CODE B$(1)>74 OR V
AL B$(2 TO )<1 OR VAL B$(2 TO )>10 THEN  GO
 TO 2105
2100 GO TO 2130
2105 LET E$=' COORDENADA INCORRECTA '
2110 LET PAP=3: LET F1=20: LET C1=3: LET FL
=1: GO SUB 370
2115 PAUSE 50: GO SUB 405
2120 IF 5-BARCO=4 THEN  GO TO 2070
2125 GO TO 2040
2130 GO SUB 490
2135 IF V$='V' THEN  GO TO 2210
2140 LET VH=0: GO SUB 185
2145 IF V=0 THEN  GO TO 2105
2150 FOR F=Y TO (Y+BARCO-1)
2155 LET C(X,F)=BARCO+1
2160 PRINT  PAPER 6; OVER 1;AT X-1,F-1;' '
2165 BEEP .1,20
2170 NEXT F
2175 GO SUB 300
2180 NEXT P
2185 FOR G=1 TO 5
2190 PRINT  PAPER 1;AT 12+G,3;'
                '
2195 NEXT G
2200 NEXT B
2205 GO TO 2250
2210 LET VH=1: GO SUB 185
2215 IF V=0 THEN  GO TO 2105
2220 FOR F=X TO (X+BARCO-1)
2225 LET C(F,Y)=BARCO+1
2230 PRINT  PAPER 6; OVER 1;AT F-1,Y-1;' '
2235 BEEP .1,20
2240 NEXT F
2245 GO TO 2175
2250 LET E$=' TODO CORRECTO '
2255 LET PAP=6: LET FL=1: LET F1=13: LET C1
=7
2260 GO SUB 370
2265 FOR N=0 TO 10
2270 BEEP .1,25: BEEP .05,20
2275 NEXT N
2280 GO SUB 405
2285 REM
2290 IF INT (RND*2) THEN  GO TO 2320
2295 LET E$=' EMPIEZAS TU '
2300 LET PAP=7: LET F1=15: LET C1=8
2305 GO SUB 370: PAUSE 50: GO SUB 405
2310 LET SPW=0
2315 GO TO 2350
2320 LET E$=' EMPIEZO YO '
2325 LET PAP=5: LET C1=8: LET F1=18
2330 GO SUB 370: BEEP .3,20: BEEP .5,15: BE
EP .3,20
2335 GO SUB 405
2340 LET SPW=1
2345 GO TO 2350
2350 REM INICIALIZACIO DE CAMPOS
2355 LET CONT=0: LET CONTSP=0
2360 LET BH=0: LET BHSP=0
2365 DIM T(12,12): LET BT=0: LET BU=0
2370 DIM O(10,10): LET T=0
2375 LET TSP=1: LET TC=0
2380 PRINT  PAPER 6;AT 12,1;'    TU    '
2385 PRINT  PAPER 2;AT 12,21;' SPECTRUM '
2390 PRINT  PAPER 5;AT 13,1;' TIROS=  ';CON
T;AT 13,21;' TIROS=  ';CONTSP
2395 PRINT  PAPER 5; BRIGHT 1;AT 14,1;' HUN
D.=  ';BHSP;AT 14,21;' HUND.=  ';BH
2400 REM EMPEZAMOS DISPAROS
2405 IF SPW THEN  GO TO 2770
2410 LET E$=' TIRAS TU '
2415 LET PAP=6: LET F1=17: LET C1=10
2420 GO SUB 370
2425 PRINT  PAPER 1;AT 15,13;'    '
2430 INPUT 'TU DISPARO ES',B$
2435 GO SUB 490
2440 GO SUB 505
2445 IF V=0 THEN  GO TO 2425
2450 GO SUB 540
2455 GO SUB 405
2460 IF V=0 THEN  GO TO 2425
2465 PRINT  PAPER 1;AT 14,16;'    '
2470 LET CONT=CONT+1
2475 PRINT AT 13,11-LEN STR$ CONT; PAPER 5;
 INK 0;CONT
2480 PRINT  FLASH 1; OVER 1;AT X-1,Y+19;' '
: PAUSE 40
2485 IF S(X,Y)>1 THEN  GO TO 2535
2490 REM AGUA
2495 PRINT  OVER 1; INK 7; FLASH 0;AT X-1,Y
+19;'*'
2500 PRINT  PAPER 5;AT 6,12;'      ';AT 8,1
2;'      ';AT 7,12;' AGUA '
2505 BEEP .5,30: BEEP .5,20
2510 FOR N=0 TO 2
2515 PRINT  PAPER 1;AT 6+N,12;'    '
2520 NEXT N
2525 LET SPW=1: GO TO 2405
2530 REM TOCADO
2535 LET BTSP=S(X,Y)
2540 IF T THEN  GO TO 2555
2545 LET T=1: DIM P(BTSP-1)
2550 DIM E(BTSP): GO TO 2540
2555 LET E(BTSP)=E(BTSP)+1: LET P(E(BTSP))=
((X-1)*100)+Y+19
2560 IF E(BTSP)=BTSP-1 THEN  GO TO 2610
2565 PRINT  PAPER 6; INK 2; OVER 1; FLASH 1
;AT X-1,Y+19;' '
2570 PRINT  PAPER 6;AT 6,12;'      ';AT 8,1
2;'      ';AT 7,12;'TOCADO'
2575 FOR N=1 TO 20
2580 BEEP .1,25
2585 NEXT N
2590 FOR N=0 TO 2
2595 PRINT  PAPER 1;AT N+6,12;'    '
2600 NEXT N
2605 GO TO 2405
2610 REM HUNDIDO
2615 LET T=0
2620 LET BHSP=BHSP+1
2625 PRINT  PAPER 6; INK 2; OVER 1; FLASH 1
;AT X-1,Y+19;' '
2630 BEEP .3,0: BEEP .3,-3
2635 INK 2: GO SUB 700: INK 9
2640 REM ADAPTACION DE E$
2645 FOR N=1 TO BTSP-1
2650 LET NUM=P(N): LET NUM1=NUM
2655 LET NUM=INT (P(N)/100)
2660 LET NUM1=NUM1-NUM*100
2665 PRINT  PAPER 2; OVER 1; BRIGHT 1;AT NU
M,NUM1;' '
2670 NEXT N
2675 PRINT  PAPER 2;AT 6,12;'      ';AT 8,
12;'      ';AT 7,12;'HUNDIDO'
2680 BEEP .8,10: PAUSE 10
2685 FOR N=0 TO 2
2690 PRINT  PAPER 1;AT N+6,12;'    '
2695 NEXT N
2700 PRINT AT 14,31-LEN STR$ BHSP; PAPER 5;
 INK 0; BRIGHT 1;BHSP
2705 IF BHSP=10 THEN  GO TO 2715
2710 GO TO 2405
2715 LET E$=' ME HAS GANADO '
2720 LET F1=18: LET C1=8
2725 LET PAP=7: GO SUB 370
2730 BEEP .5,0: PAUSE 50: GO SUB 405
2735 LET E$=' ECHAMOS OTRA? ': LET C1=8
2740 GO SUB 370
2745 LET A$=INKEY$
2750 IF A$='' THEN  GO TO 2745
2755 IF A$='S' THEN  RUN
2760 CLS : GO TO 10000
2765 REM DISPARA SPECTRUM
2770 GO SUB 940
2775 IF BT=1 THEN  LET X=XU: LET Y=YU: GO T
O 2855
2780 LET BCT=0
2785 GO SUB 975
2790 GO SUB 890
2795 GO SUB 855
2800 GO SUB 780
2805 IF V=1 THEN  GO TO 2795
2810 LET C1=10: LET F1=16: LET E$='
    ': GO SUB 405
2815 IF C$='A' THEN  GO TO 2830
2820 IF C$='T' THEN  GO SUB 1050: GO TO 285
0
2825 IF C$='H' THEN  GO SUB 1050: INK 6: GO
 SUB 695: INK 9: GO TO 2870
2830 GO SUB 1115
2835 PRINT  PAPER 1;AT 18,14;'    '
2840 LET SPW=0
2845 GO TO 2405
2850 LET BT=1:: DIM D(4)
2855 GO SUB 1125
2860 IF SPW=0 THEN  GO TO 2405
2865 IF BU=0 THEN  GO TO 2405
2870 GO SUB 1070
2875 LET BU=0: LET BH=BH+1
2880 LET BT=0
2885 INK 6: GO SUB 695: INK 9
2890 PRINT  PAPER 1;AT 18,14;'    '
2895 PRINT  PAPER 1;AT 6,12;'    '
2900 PRINT AT 14,11-LEN STR$ BH; BRIGHT 1;
PAPER 5; INK 0;BH
2905 IF BH<10 THEN  GO TO 2405
2910 LET E$=' TE HE GANADO ': LET PAP=7: LE
T FL=1
2915 LET F1=17: LET C1=9: GO SUB 370
2920 PAUSE 100
2925 GO SUB 405
2930 LET E$=' ECHAMOS OTRA? '
2935 LET FL=0: LET C1=8
2940 GO SUB 370
2945 IF INKEY$='' THEN  GO TO 2945
2950 LET K$=INKEY$
2955 IF K$='S' THEN  RUN
2960 IF K$='N' THEN  GO TO 10000
2965 GO TO 2945
2970 LET C=0
2975 DATA .5,5,.5,4,.5,2,.5,0,.2,9,.08,9,.0
8,9,.2,9
2980 DATA .25,0,.7,9,.3,-59,.5,5,.5,4,.5,5,
.5,2,.2,10,.08,10
2985 DATA .08,10,.2,10,.25,2,.7,10,.3,-59,.
35,7,.1,6,.35,7,.1,6
2990 DATA .35,7,.1,6,.20,7,.2,10,.2,9,.08,9
,.08,7,.1,9,.1,-59
2995 DATA .1,10,.1,-59,.2,12,.2,-1,.2,0,.2,
1,.2,2,.08,2,.08,1
3000 DATA .2,2,.25,5,.2,4,.1,4,.1,3,.1,4,.1
,-59,.2,7,.3,5
3005 DATA .2,-59,.3,0,.2,-59,.3,5
3010 RESTORE 2975
3015 FOR n=1 TO 57
3020 READ A,B
3025 BEEP A,B
3030 NEXT N
3040 RUN
```



La pantalla de juego representa el tablero de control del puente de mando, desde el que podemos dirigir el ataque de nuestra flota.

DOMINANDO EL TECLADO

El modo **C** es el siguiente tipo de cursor que estudiaremos. Para pasar el cursor a este modo, debemos pulsar **CAPS SHIFT** y **2**. Estas teclas funcionan como un interruptor, de manera que nos permiten cambiar del cursor **L** al **C** y viceversa. El modo **C** tiene, por lo tanto, un efecto muy similar al de la tecla de bloqueo de mayúsculas en una máquina de escribir convencional **(CAPS LOCK)**.

Durante la introducción de un texto extenso que emplea mayúsculas, deberíamos tener pulsada contínuamente la tecla **CAPS SHIFT**; gracias al modo **C** podremos evitar este inconveniente. Todo texto que introduzcamos con este cursor, será presentado automáticamente en mayúsculas.

Cuando estando en el modo **K** pulsamos **CAPS LOCK (CAPS SHIFT** y **2)**, no se aprecia inmediatamente el cambio de modo. Sin embargo, el ordenador recordará este hecho, y el cursor cambiará a modo **C** una vez introducida la palabra clave. El retorno al modo **L** se conseguirá pulsando nuevamente las teclas **CAPS SHIFT** y **2**.

EL MODO G

Este es el modo que nos permite la introducción de los caracteres gráficos presentes en el teclado. Para acceder a él, pulsaremos las teclas **CAPS SHIFT** y **9** a un mismo tiempo. Inmediatamente, el cursor pasará a representarse como una **G**, y mediante la pulsación de las teclas **1** al **8** obtendremos el carácter gráfico correspondiente a la tecla pulsada, que aparece serigrafiado en color blanco sobre ella. Para abandonar el modo **G** sólo es necesario volver a pulsar la tecla del **9**, sin que sea esta vez imprescindible la pulsación conjunta de **CAPS SHIFT**.

Gracias a los gráficos predefinidos podemos realizar dibujos de relativa precisión. Son los denominados «dibujos en baja resolución».

¡!

El paso entre el modo **K** y cualquiera de los otros cuatro (**L, C, E** y **G**), se efectúa de manera automática por el ordenador.

El cambio de cursores entre los modos distintos de **K**, se efectúa mediante una acción sobre alguna tecla de control.

Los gráficos antes mencionados se denominan PREDEFINIDOS, y gracias a ellos podremos construir dibujos para nuestros propios programas. Para explicar el correcto sistema de introducción de los gráficos predefinidos, vamos a utilizar un ejemplo. Supongamos que disponemos de un cuadrado de las dimensiones de un carácter, totalmente macizo.

Como podemos observar en el teclado, los gráficos predefinidos son combinaciones de cuartas partes de nuestro imaginario cuadrado.

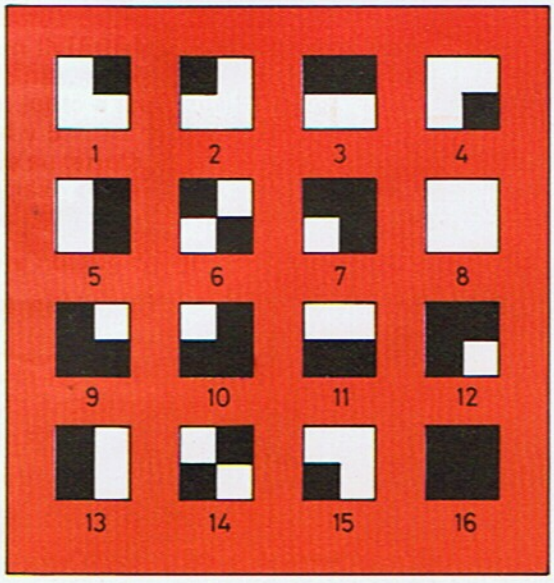
Pues bien, si obtuviéramos todas las combinaciones posibles de las cuartas partes del cuadrado original, descubriríamos que existen 16 formas diferentes. Sin embargo, el número de teclas destinadas a tal efecto es sólo ocho.

¿A qué es debido esto? ¿Es que sólo se puede representar ocho de las dieciséis combinaciones posibles? No, ocho de las mismas se obtienen mediante la pulsación directa en modo **G** de las teclas numéricas; las restantes precisan además la pulsación simultánea de la tecla **CAPS SHIFT.** Cada tecla numérica, tiene serigrafiado en blanco una combinación de cuartos de carácter. En el teclado del Spectrum Plus, cuando se pulsa la tecla sin **CAPS SHIFT,** la forma representada corresponde a dicha combinación. Por el contrario, cuando se pulsa la tecla juntamente con **CAPS SHIFT,** el carácter representado es la combinación formada por el resto del cuadrado, es decir, la parte del color de fondo de la tecla.

En el otro modelo de Spectrum, el significado es justamente el contrario. Los gráficos pulsados con **CAPS SHIFT** visualizan la combinación serigrafiada en blanco sobre la tecla, mientras que los pulsados sin **CAPS SHIFT** producen la combinación complementaria (similar a un negativo fotográfico de la combinación anterior).

EL MODO E

Por último, hablaremos del modo EXTENDIDO, que se representa con el cursor **E.** A él se accede mediante la pulsación simultánea de **CAPS SHIFT** y **SYMBOL SHIFT,** bastando con volver a pulsar estas mismas teclas para sa ir del modo. Con el cursor en el modo *extender* o extendido, se tiene acceso a la mayor parte de las funciones científicas y de programación. Es decir, refiriéndonos al teclado estándar, todas las serigrafiadas en verde y rojo sobre la misma base del teclado. Con el cursor en el modo **E** y la sola pulsación de una tecla, se accede a la función marcada en color verde (encima de la tecla). Por ejemplo, pulsando **C** obtenemos **LPRINT,** y pulsando **O, PEEK.**



El número total de gráficos PREDEFINIDOS es 16. Se obtienen mediante la pulsación de las teclas numéricas 1 a 8, en modo G (con o sin CAPS SHIFT).

En el mismo modo, pero manteniendo pulsada la tecla **SYMBOL SHIFT**, se accede a las funciones serigrafiadas en rojo sobre la base del teclado. Volviendo a los ejemplos anteriores, **PAPER** y **OUT**, respectivamente.

LOS CARACTERES DE CONTROL DE LA FILA SUPERIOR

Ya hemos hablado anteriormente, de dos de las teclas de control que se encuentran en la fila superior del teclado (la fila de los números): la función **DELETE (0)** o borrado, y la función **CAPS LOCK (2)** o bloqueo de mayúsculas. También sabemos que a las teclas de control se tiene acceso con la pulsación de **CAPS SHIFT** y el número correspondiente del **1** al **0**. Vamos a estudiar ahora el resto de los caracteres de control de la fila superior, que no hemos visto hasta el momento.

La tecla **1** corresponde a la función **EDIT**, que describiremos más adelante, a la hora de enfrentarnos con la introducción y corrección de nuestro primer programa en BASIC.

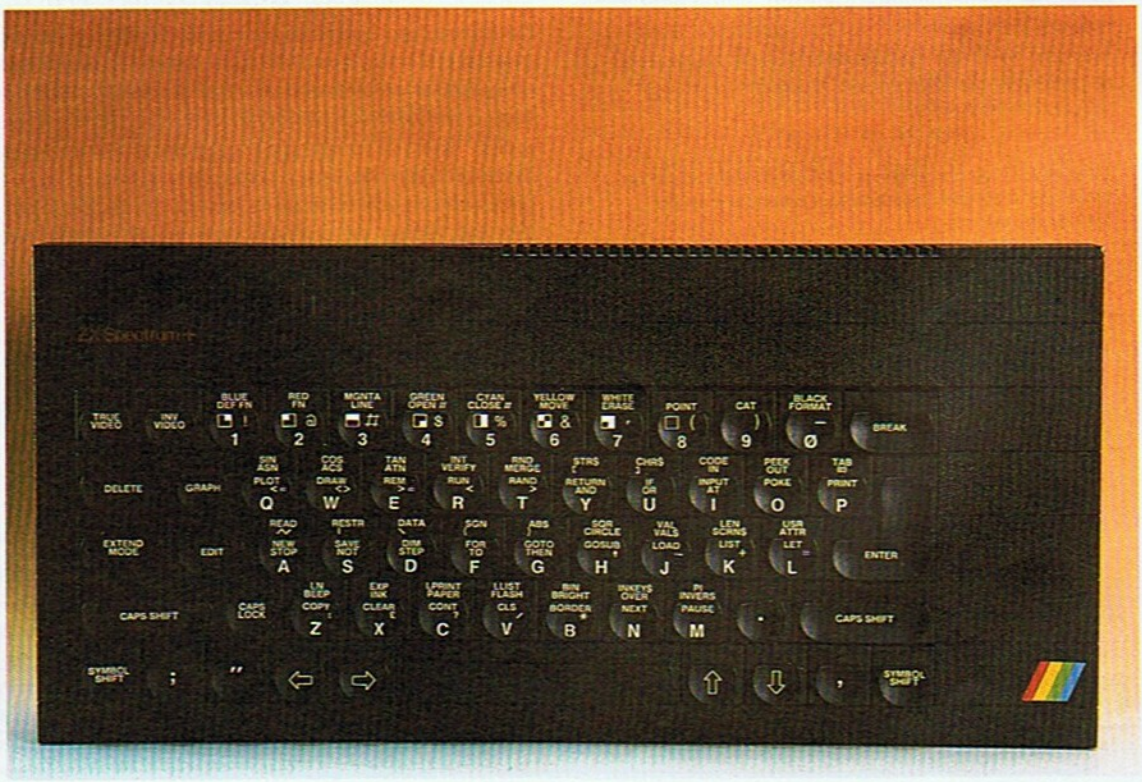
Las teclas **3** y **4** son, respectivamente, las funciones **TRUE VIDEO** e **INVERSE VIDEO**. Para explicar, aunque sin entrar en detalles, el empleo de estas funciones, debemos conocer algunas cosas sobre la forma en que el Spectrum es capaz de mostrar caracteres en la pantalla. Existen tres tipos diferentes de representación de caracteres.

La primera de ellas es el vídeo normal, en inglés *true video,* que consiste en mostrar cada carácter en color negro sobre fondo blanco. El segundo sistema es el de vídeo inverso, en inglés *inverse video,* que representa el carácter en blanco sobre fondo negro (como un negativo fotográfico). Por último, nos queda comentar el tercer modo, al que llamamos parpadeante, en inglés *flash,* que se utiliza siempre en la representación del cursor.

Este último sistema es una combinación de los dos anteriores. El ordenador muestra el texto en vídeo normal y vídeo inverso alternativamente, con lo que se consigue un efecto de resalte de lo escrito.

De momento debe bastarnos con saber, que por medio de las funciones **TRUE VIDEO** e **INVERSE VIDEO** (teclas **3** y **4**), podemos conseguir la representación en pantalla de caracteres y símbolos gráficos de los dos primeros tipos, es decir,

En el teclado del Plus existe un mayor número de teclas que proporciona una considerable comodidad de manejo.



¡!

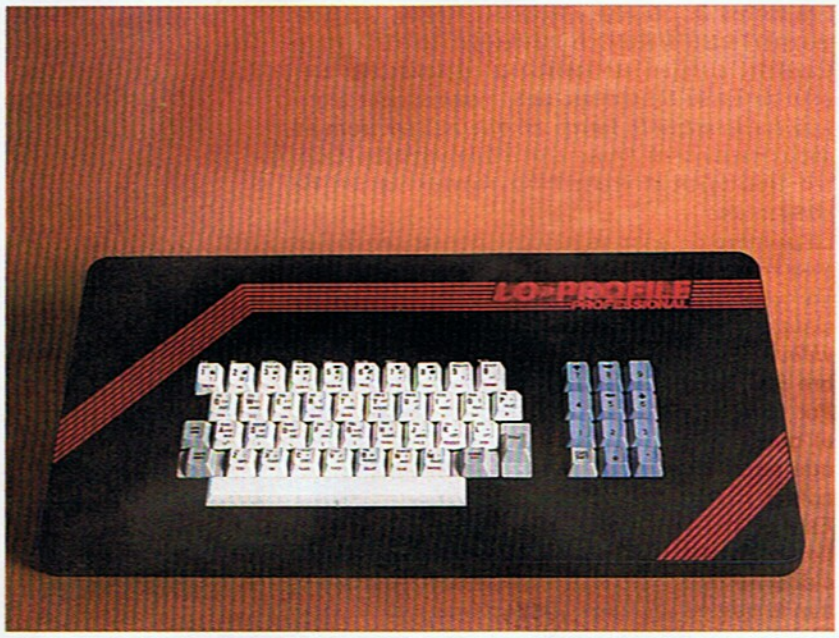
El paso al modo **C**, así como su abandono, se consigue pulsando a un tiempo las teclas **CAPS SHIFT** y **0**.
El paso al modo **E**, así como su abandono, se consigue pulsando a un tiempo las teclas **CAPS SHIFT** y **SYMBOL SHIFT**.

*

La función **CAPS LOCK**, se consigue mediante la pulsación de **CAPS SHIFT** y **2**, simultáneamente. Su misión es la misma, que la de la tecla de bloqueo de mayúsculas de una máquina de escribir convencional.

*

La tecla de control **BREAK** se emplea únicamente para detener la ejecución de programas. Si se utiliza en el modo directo, no tendrá otro efecto que el de escribir un carácter «espacio».

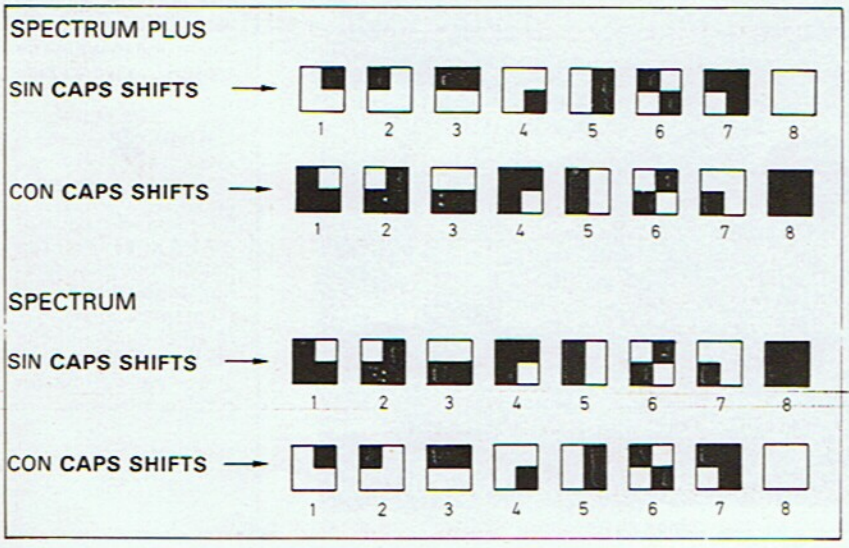


Numerosos fabricantes de hardware *han lanzado al mercado teclados profesionales con los que suplir las deficiencias existentes en el teclado original del Spectrum.*

en vídeo normal e invertido; aunque para sacar partido del uso de estas funciones, deberemos esperar aún un poquito más a profundizar en conocimientos.

Los caracteres de control **5** y **8** desplazan el cursor, un carácter o palabra clave a la izquierda y derecha, respectivamente. Empleando este movimiento horizontal del cursor, podemos desplazarnos por la línea donde escribimos caracteres en la pantalla; de este modo, podemos corregir errores mecanográficos o efectuar inclusiones de caracteres, además de borrados, en combinación con las teclas **CAPS SHIFT** y **0 (DELETE)**. La misión de las teclas de control **6** y **7** la comentaremos más adelante, a la hora de introducir nuestro primer programa, ya que dichas teclas no nos son útiles cuando trabajamos en modo directo. No debemos de confundir el objetivo de estas dos funciones con el del movimiento del cursor en horizontal, aunque físicamente se parezcan.

La serigrafía de los gráficos PREDEFINIDOS en las teclas del Plus está invertida con respecto a la del Spectrum normal.



EL TECLADO DEL SPECTRUM PLUS

Hemos dado un completo paseo por el teclado para familiarizarnos con él y aprender a obtener cada una de las palabras clave, símbolos, letras, caracteres gráficos y números que lo componen, así como algunas funciones, aunque sin entrar en el detalle de su empleo.

En atención a los usuarios del modelo Plus debemos decir que, aunque en esencia los teclados de éste y el modelo antiguo son muy similares, existen algunas mejoras en el Plus en cuanto al tacto de las teclas y la forma de obtener algunas funciones y símbolos de uso habitual.

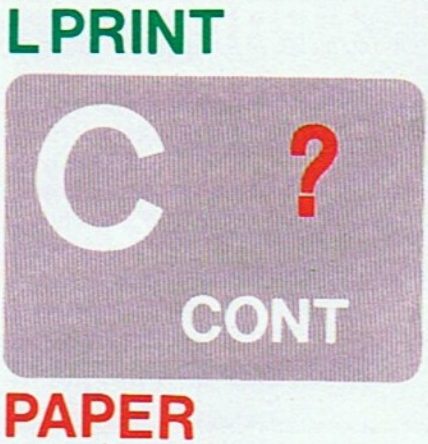
Comenzamos con el pulsador de *RESET* o arranque en caliente, que realiza la misma función que el apagado y vuelta a encender del ordenador por medio de la desconexión y conexión de la clavija que lo une a la fuente de alimentación, con la evidente ventaja de una mayor comodidad.

Otra mejora consiste en la independización de todas las teclas de control, que en el modelo estándar se obtienen mediante la pulsación de **CAPS SHIFT** y alguna otra tecla. Así, en el Spectrum Plus, mediante la pulsación de una sola tecla se obtienen las funciones:

TRUE VIDEO, INVERSE VIDEO, DELETE, GRAPHICS, EDIT, CAPS LOCK, EXTENDED y BREAK

A todas ellas se accede por pulsación directa, sin necesidad de ningún otro **SHIFT**.

La tecla **DELETE** cumple la misma función que **CAPS SHIFT** y **0** en el modelo estándar. La tecla **EDIT** permite el acceso directo a esta función sin tener que pulsar **CAPS SHIFT** y **1**. La tecla **CAPS LOCK**, o bloqueo de mayúsculas, efectúa la misma función que **CAPS SHIFT** y **2** en el mo-



Las funciones que el Spectrum ofrece serigrafiadas en verde y rojo sobre la placa del teclado, se obtienen con el cursor en modo E.

Los dos tipos principales de impresión en pantalla son el vídeo normal y el vídeo invertido. Existe un tercer tipo, denominado parpadeante, que se consigue mediante el empleo alternativo de los dos modos principales.

El tecleo del Plus conlleva una mayor comodidad de uso, ya que algunas funciones que en el otro modelo se obtenían mediante la pulsación de dos teclas, se pueden obtener en el Plus de manera directa.

delo antiguo. La tecla **EXTENDED** pasa el cursor directamente a este modo, siendo su correspondencia en el otro modelo **CAPS SHIFT** y **SYMBOL SHIFT** simultáneamente. La tecla **GRAPHICS** efectúa las veces de **CAPS SHIFT** y **9** en el modelo convencional. Terminando con la parte izquierda del teclado, se encuentran **TRUE VIDEO** e **INVERSE VIDEO**, que emulan las funciones que en el modelo estándar proporcionan **CAPS SHIFT** y **3** para el primer caso, y **CAPS SHIFT** y **4** para el segundo.

Las teclas de desplazamiento del cursor se encuentran en el Plus a la izquierda y derecha de la barra espaciadora, que cumple las veces de la tecla **SPACE** del modelo estándar. Las funciones de estas teclas de movimiento, vienen a sustituir a las que en el otro modelo se obtienen pulsando **CAPS SHIFT** y las teclas del **5** al **8**.

Pero además de estas teclas de función, existen otros símbolos en el Spectrum convencional a los que se accede a través de **SYMBOL SHIFT**, y que en el modelo Plus tienen la comodidad de ser ejecutables con una sola pulsación.

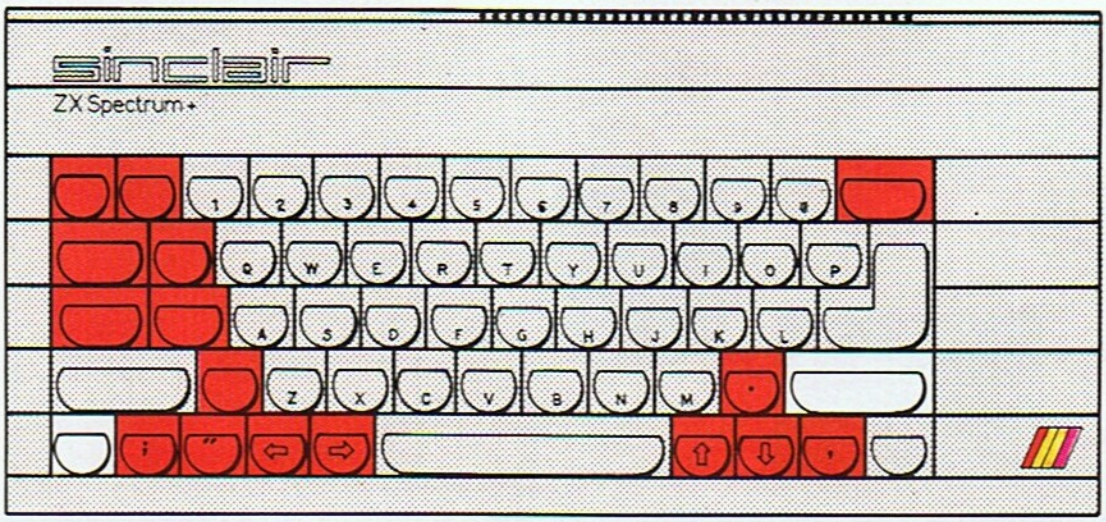
Estas teclas son los caracteres de puntuación (el punto, la coma y el punto y coma) y las comillas, situados todos en la línea inferior del teclado, a ambos lados de la barra espaciadora.

También, en el teclado del PLUS y para una mayor comodidad de operación, están duplicadas las teclas de **CAPS SHIFT** y **SYMBOL SHIFT** a cada lado del teclado, de forma que pueden pulsarse con la mano izquierda o derecha indistintamente.

Otra de las diferencias importantes existentes entre los teclados del modelo antiguo y del Plus, es que en el primero la serigrafía del teclado es a ocho colores, facilitando la identificación de qué tipo de **SHIFT** pulsar para acceder a un determinado símbolo o función; mientras que en el segundo, toda la serigrafía es en color blanco, debiendo guiarnos únicamente por el orden de co-

Las teclas de control **6** y **7**, correspondientes a los desplazamientos verticales del cursor, no ejercen ningún efecto sobre la línea inferior de pantalla, en la que indicamos las órdenes al Spectrum.

El paso al modo **G**, se consigue pulsando a un tiempo las teclas **CAPS SHIFT** y **9**. Para su abandono, sólo es necesaria una nueva pulsación de la tecla **9**.



locación de los rótulos dibujados sobre cada tecla.

Para establecer una relación de identificación de funciones en el teclado del Spectrum Plus, podemos decir que de las tres líneas serigrafiadas sobre las propias teclas, la primera se corresponde con la palabra clave, la segunda con las palabras y símbolos serigrafiados en rojo sobre las teclas del modelo antiguo, y la tercera con la letra o número de la tecla en sí.

Por otra parte, lo que se obtiene con el teclado en modo extendido se encuentra serigrafiado sobre la base de las teclas en dos líneas; la primera se corresponde con la serigrafía en color verde del modelo estándar y la segunda con la serigrafía en rojo.

En este estado de cosas, para tener acceso a la segunda línea de estas funciones, además de pulsar la tecla de **EXTENDED** es necesario mantener pulsada una de las de **SYMBOL SHIFT**.

Es importante saber que existen determinadas funciones que en el modelo antiguo sólo se pueden obtener mediante la pulsación de dos teclas y que en el Plus, además de disponer de una tecla independiente, también pueden conseguirse de la misma manera.

Veamos un ejemplo. En el modelo antiguo, la función de borrado **(DELETE)** se consigue con la acción simultánea sobre las teclas **CAPS SHIFT** y **O**. El Spectrum Plus, dispone por el contrario de una tecla destinada a tal fin, denominada **DELETE**. Pues bien, a pesar de esto último, la pulsación simultánea de las teclas **CAPS SHIFT** y **0** en el modelo Plus, ejecutará también una acción de borrado.

Por tanto, podemos decir que el teclado de un modelo Plus simula a la perfección el de un Spectrum antiguo, aunque además incorpora mejoras con respecto a éste.

Mediante el cursor E se tiene acceso a las dos palabras serigrafiadas sobre las teclas del Spectrum Plus.

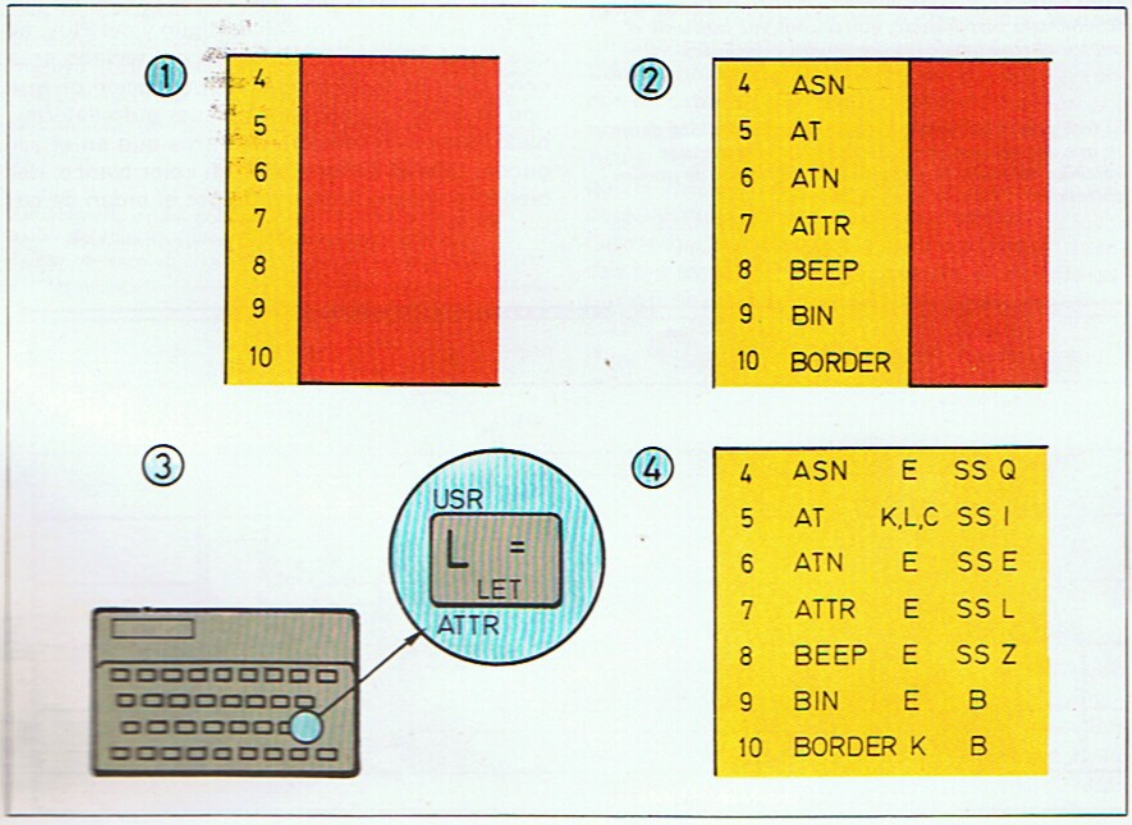
¡!

El teclado del Spectrum Plus, simula un teclado de Spectrum convencional, incorporándole además algunas ventajas.

*

El pulsador de *RESET* del Spectrum Plus, facilita el apagado y encendido del ordenador sin necesidad de un corte en la alimentación.

Estos son los pasos a seguir para la resolución del ejercicio práctico.



EJERCICIO DE RECAPITULACION

Al llegar a este punto, ya somos capaces de interpretar perfectamente el teclado de nuestro Spectrum. Ha llegado el momento de que pongamos a prueba nuestro conocimientos mediante un sencillo ejercicio. Para ello utilizaremos la tabla que aparece a continuación; en ella están representados algunos de los caracteres que se pueden obtener a través del teclado, así como diversas funciones que no suponen la impresión de un carácter concreto.

Todos estos caracteres o funciones, tienen un número de orden a su izquierda; y a su derecha, la manera por la que se obtienen en el teclado. Para indicar el sistema de edición del carácter o función en cuestión, hemos empleado la siguiente notación: en primer lugar figura el cursor mediante el cual se pueden obtener, y a continuación la tecla o teclas que es necesario pulsar para su edición.

El ejercicio práctico consistirá en lo siguiente: una vez tapadas las dos últimas columnas de la mencionada tabla con una hoja de papel, elegiremos al azar un número de los existentes en la columna izquierda de la tabla. A continuación, correremos el papel hacia la derecha, dejando al descubierto la segunda columna. Una vez hecho esto, deberemos intentar averiguar, cual es la forma correcta de obtener el carácter o función que aparezca en esta segunda columna, en la fila del número elegido.

Como es evidente, la solución al ejercicio figura en la tercera columna, que por tanto deberá permanecer tapada hasta que nos decidamos por una posible solución a la cuestión planteada. Este ejercicio habremos de practicarlo todas las veces que sea necesario, hasta obtener el porcentaje de aciertos deseable.

Ahora bien, para la resolución del ejercicio contaremos con la inestimable ayuda de un gran amigo: el Spectrum. Efectivamente, podremos utilizar el ordenador para guiarnos en la obtención de soluciones y ¡cómo no! para comprobarlas antes de resignarnos a destapar la fatal columna izquierda.

TABLA PARA EL EJERCICIO DEL TECLADO

Núm.	Carácter/Función	Modo de obtención	
1	ABS	E	G
2	ACS	E	SS W
3	AND	K, L, C	SS Y
4	ASN	E	SS Q
5	AT	K, L, C	SS I
6	ATN	E	SS E
7	ATTR	E	SS L
8	BEEP	E	SS Z
9	BIN	E	B
10	BORDER	K	B
11	BRIGHT	E	SS B
12	CAPS LOCKT	K, L	CS 2*
13	CAT	E	SS 9
14	CHR$	E	U
15	CIRCLE	E	SS H
16	CLEAR	K	X
17	CLOSE #	E	SS 5
18	CLS	K	V
19	CODE	E	I
20	CONTINUE (CONT)	K	C
21	COPY	K	Z
22	COS	E	W
23	DATA	E	D
24	DEF FN	E	SS 1
25	DELETE	K, L, C	CS 0*
26	DIM	K	D
27	DRAW	K	W
28	EDIT	K, L, C	CS 1*
29	ERASE	E	SS 7
30	EXP	E	X
31	FLASH	E	SS V
32	FN	E	SS 2
33	FOR	K	F
34	FORMAT	E	SS 0
35	GOSUB	K	H
36	GOTO	K	G
37	GRAPHICS	K, L, C	CS 9*
38	IF	K	U
39	IN	E	SS I
40	INK	E	SS X
41	INKEY$	E	N
42	INPUT	K	I
43	INT	E	R
44	INVERSE	E	SS M
45	INVERSE VIDEO	K, L, C	CS 4*
46	LEN	E	K
47	LET	K	L
48	LINE	E	SS 3
49	LIST	K	K
50	LLIST	E	V
51	LN	E	Z
52	LOAD	K	J
53	LPRINT	E	C
54	MERGE	E	SS T
55	MOVE	E	SS 6
56	NEW	K	A
57	NEXT	K	N
58	NOT	K, L, C	SS S
59	OPEN #	E	SS 4
60	OR	K, L, C	SS U
61	OUT-	E	SS O
62	OVER	E	SS N
63	PAPER	E	SS C
64	PAUSE	K	M
65	PEEK	E	O
66	PI	E	M
67	PLOT	K	Q
68	POINT	E	SS 8
69	POKE	K	O
70	PRINT	K	P
71	RANDOMIZE (RAND)	K	T
72	READ	E	A
73	REM	K	E
74	RESTORE	E	S
75	RETURN	K	Y
76	RND	E	T
77	RUN	K	R
78	SAVE	K	S
79	SCREEN$	E	SS K
80	SGN	E	F
81	SIN	E	Q
82	SQR	E	H
83	STEP	K, L, C	SS D
84	STOP	K, L, C	SS A
85	STR$	E	Y
86	TAB	E	P
87	TAN	E	E
88	THEN	K, L, C	SS G
89	TRUE VIDEO	K, L, C	CS 3*
90	TO	K, L, C	SS F
91	USR	E	L
92	VAL	E	J
93	VAL$	E	SS J
94	VERIFY	E	SS R

CS CAPS SHIFT
SS SYMBOL SHIFT
* Estas teclas están independientes en el Plus.

PROTECCION DEL SOFTWARE

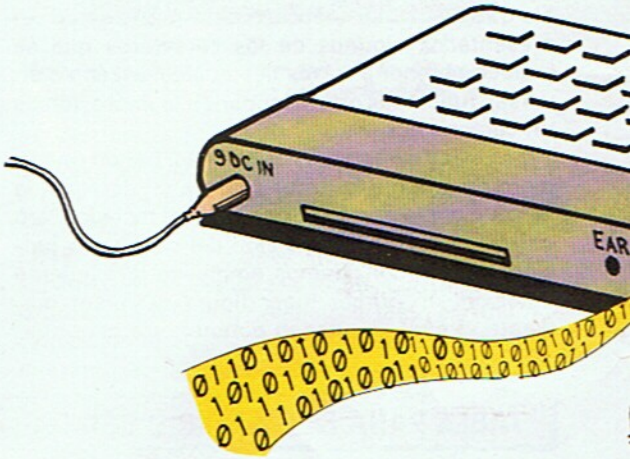
Sin lugar a dudas, existen muy diversas técnicas para conservar nuestros datos y programas. A veces, ese programa que tanto tiempo nos ha costado realizar, se resiste a «entrar» de nuevo en el ordenador. En otras ocasiones, nos gustaría mantener oculta esa rutina que con menos bytes realiza el mismo trabajo... y en menos tiempo. Quizás tan solo pretendamos sorprender a aquellos que intentan inmiscuirse en alguno de nuestros listados.

A continuación, desde distintos puntos de vista, desarrollaremos algunas técnicas para solucionar estos problemas, lo que facilitará la protección de nuestros programas, y nos ahorrará quebraderos de cabeza a la hora de intentar cargarlos.

Cuando grabamos un programa para su posterior utilización, lo hacemos mediante el comando **SAVE.** Al pulsar **ENTER** se emite en la parte inferior de la pantalla el mensaje *Start tape then press any key* (pon en marcha la cinta, luego presiona cualquier tecla). Una vez hecho esto, aparecen en la pantalla las clásicas bandas de comienzo de grabación y, finalmente, si todo ha ido bien, el informe **OK.** Sin embargo, no podemos todavía estar seguros de que la copia esté almacenada correctamente en la cinta.

Si hemos seguido todos los pasos que el manual del Spectrum indica, no existen malas conexiones entre ordenador y grabadora, y ésta se encuentra en buenas condiciones al igual que la cinta magnética, es posible que finalmente la grabación haya quedado perfecta.

Al efectuar una grabación de datos en casete, hemos de tener mucho cuidado y no olvidar conectar la clavija MIC de la grabadora.

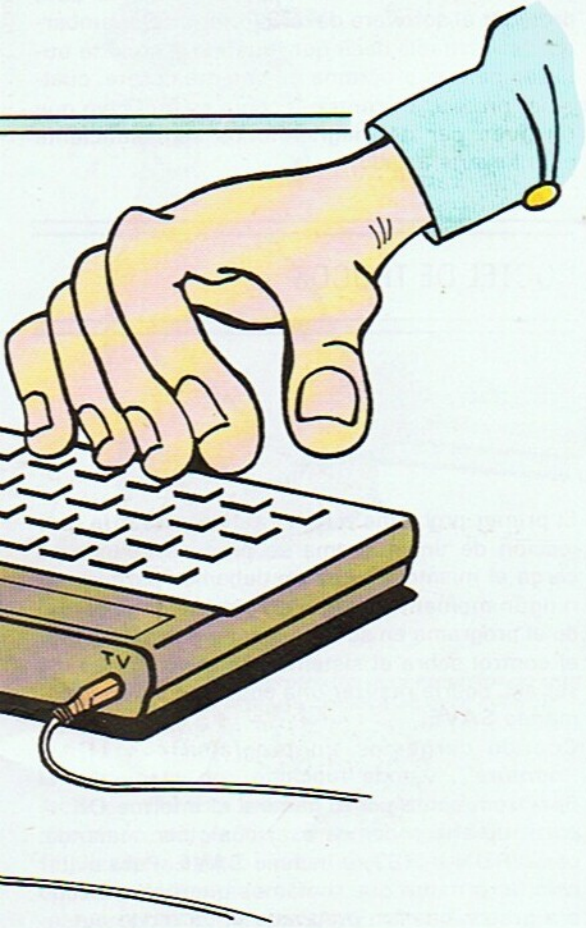


Como vemos, pueden ser muchas las causas que den lugar a errores en la conservación de programas. Es más, si por olvido, simplemente no hemos conectado la salida MIC del Spectrum a la del casete, aparentemente todo irá bien, pero estaremos «llenando» nuestra habitación de bits, y no la cinta de grabación.

Si aún no hemos desenchufado el ordenador y seguimos conservando en memoria el programa, todavía no ha ocurrido ninguna «desgracia» irreparable y podemos repetir, ahora correctamente, el proceso de grabación. En caso contrario, no tendremos otro remedio que volver a reconstruir paso a paso nuestro programa.

Por todo ello, sea el que sea el medio de almacenamiento que estemos utilizando (casete, microdrive o disco), después del **SAVE** debemos realizar la comprobación de que la información conservada es la misma que la que se encuentra en la memoria del ordenador. El comando **VERIFY** realiza esta función; si una vez terminada la verificación el Spectrum emite el informe **OK,** habrá una garantía más de que nuestro trabajo no se ha «esfumado» sin dejar ni rastro. Pese a

Algunos fabricantes de **hardware**, *diseñan dispositivos para proteger a los programadores del tan temido «apagón».*

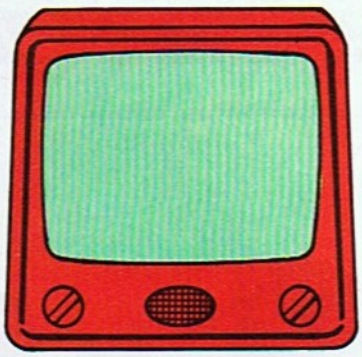


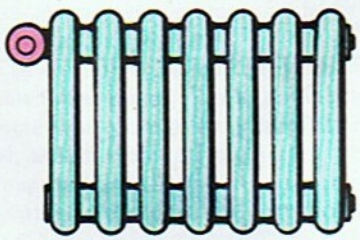
todo, hemos de advertir que, como veremos más adelante, habrá ocasiones en que el comando **VERIFY** no será tan útil como nosotros quisiéramos.

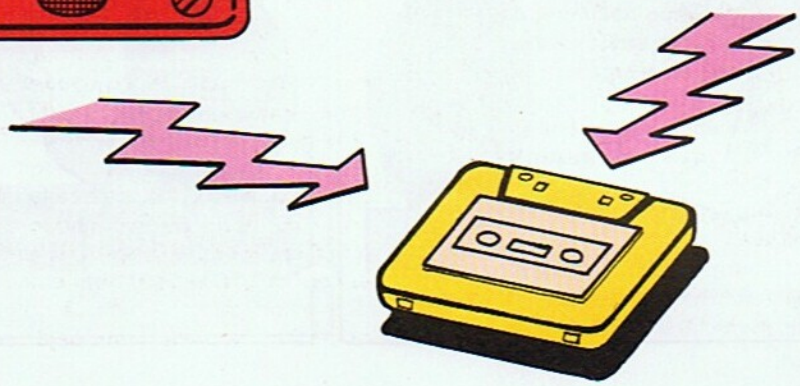
Para evitar todos estos inconvenientes, es necesario disponer de copias de seguridad de aquellos programas que más utilicemos. Para este tipo de software algunas firmas inglesas comercializan memorias **EPROM** *(Erasable Programmable Read Only Memory),* que además de proporcionar una mayor fiabilidad de los datos almacenados, facilitan una velocidad de carga superior a la de los métodos convencionales.

Existen también dispositivos de protección del software contra el tan temido «apagón», con la consiguiente pérdida de toda la información almacenada en la memoria. Estos aparatos, basados en unas pequeñas baterías, suministran durante aproximadamente 15 minutos la energía necesaria para que el ordenador siga funcionando. Gracias a ello, podremos realizar una copia del trabajo realizado, a la vez de alejar el deseo de cometer un atentado terrorista contra la compañía de la luz.

Hay que tener cuidado de alejar los soportes magnéticos, de cualquier fuente de irradiación magnética o calor intenso.







BITS

De entre las muchas maneras de inicializar el sistema mediante llamadas a la R.O.M., unas son más espectaculares que otras. He aquí una nueva muestra de ello:
RANDOMIZE USR 5050

*

Algunas simples llamadas a subrutinas de la R.O.M. pueden producir efectos curiosísimos. Este es el caso de **RANDOMIZE USR 4710.**

PROTECCION CONTRA EL AMIGO DE LO AJENO

Hemos hablado de la inevitable necesidad de disponer de copias de nuestros programas. Incluso con una cinta recién adquirida, corremos el riesgo de perder la información que contiene; simplemente, situándola algún tiempo cerca de un televisor, o de cualquier otra fuente magnética o de calor intenso.

Seguramente, nos estaremos preguntando cómo realizar las copias de seguridad de las cintas, ya que la mayoría de los programas vienen protegidos de una u otra manera. Quizás queremos pasarlos al microdrive para evitar la interminable espera cada vez que deseamos cargarlos; o tal vez, seamos nosotros mismos los que deseamos proteger nuestros propios programas.

Lo cierto es que los fabricantes de software se esmeran cada vez más en mantener cerrados a los ojos del usuario las interioridades de sus productos. Esto se debe, principalmente, a que la propiedad intelectual sobre programas se encuentra bastante desprotegida, y el creciente número de «piratas» que lanzan copias ilegales al mercado, sin ningún escrúpulo, lesiona gravemente los intereses de los que con tantas horas de trabajo crearon el programa.

Por otra parte, existe la idea de que un programa, si no está bien protegido, no puede ser muy bueno; aunque, evidentemente, esta relación de correspondencia no se cumple en la mayoría de los casos.

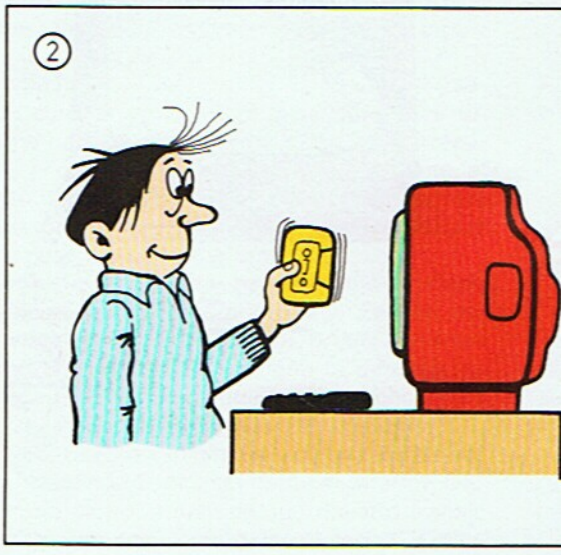
A continuación vamos a ver una serie de trucos y técnicas, que se utilizan habitualmente para proteger el software de la «piratería». Sin embargo, es inevitable decir que, cuando el soporte utilizado para el programa es la cinta casete, cualquier protección contra la copia es inútil, ya que un buen par de magnetófonos será suficiente para llevarla a cabo.

COCTEL DE TRUCOS

El primer problema a la hora de efectuar la protección de un programa se produce cuando se carga el mismo. Lo que no debemos permitir en ningún momento, es que el «pirata» tenga cargado el programa en su ordenador y pueda recobrar el control sobre el sistema operativo ya que, de ser así, podría realizar una copia mediante el comando **SAVE.**

Cuando cargamos un programa con **LOAD** *''nombre''*, y todo funciona correctamente, el Spectrum emite por lo general el informe **OK.** A partir de ahí, podemos usar cualquier comando, como **RUN** o **LIST**, e incluso **SAVE.** Para evitar esto, lo primero que teníamos que haber hecho era grabar nuestro programa en el modo autoe-

Si bien el intercambio de programas entre aficionados se puede considerar ocasionalmente lícito, lo que todos debemos rechazar es la intención de algunos individuos sin escrúpulos, de hacer negocio con el esfuerzo y el trabajo ajeno, mediante la venta ilegal de programas.



Aunque los métodos de protección aquí expuestos no sean infalibles, al menos le proporcionarán un buen dolor de cabeza al «pirata».

jecución. De esta manera, el control del ordenador pasará automáticamente al programa una vez finalizada la carga, sin dar oportunidad al «copiador de programas» a utilizar su herramienta más elemental: el comando **SAVE.**

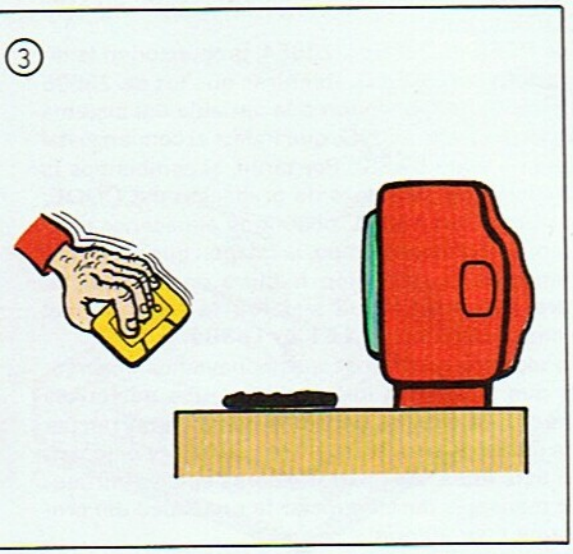
Para utilizar el sistema de autoejecución, es necesario grabar el programa mediante *SAVE* ***"nombre"*** **LINE** *xxxx;* donde *xxxx,* es el número de línea de programa por la que deseamos comience la ejecución automática. Pero optando únicamente por esta solución, sólo habremos demorado la aparición del problema, ya que aunque el programa se autoejecute, el «pirata» podrá interrumpirlo mediante **BREAK,** recuperando de nuevo el control de la situación.

El próximo problema a resolver es, por tanto, la fácil interrupción de nuestro programa mediante **BREAK.** Las soluciones a buscar son evidentemente de dos tipos, y ambos centrados en la vulnerabilidad residente en **BREAK.** Una de ellas, es conseguir que la tecla **BREAK** tenga un comportamiento distinto del habitual, que como sabemos es la detención del programa con la emisión de un mensaje de error. El otro tipo de solución, consiste en inhibir la tecla en cuestión, dejando sin ningún efecto su pulsación. Estudiaremos a continuación algunos ejemplos concretos de los dos sistemas expuestos.

Los trucos para el cambio de efecto en la tecla **BREAK,** se basan en el manejo de algunas variables del sistema que contienen información esencial para el sistema operativo, si bien no afectan de forma ineludible al correcto funcionamiento de un programa. Precisando un poco más, los trucos de este tipo no se basan en que **BREAK** no realice la detención por error, sino que no se recobre el control sobre el sistema o, de ser así, que ello ocurra después de la destrucción de la información en memoria.

El primero de los ejemplos de este tipo altera la variable del sistema conocida como **DF SZ,** que controla el número de líneas de la parte inferior de la pantalla, es decir, las reservadas al sistema operativo. Usualmente, el valor contenido en esta variable es 2, que es el número mínimo de líneas que el sistema requiere para la impresion de un mensaje (una para el mensaje y otra en

③



BITS

Existen pequeñas subrutinas BASIC con las que podemos conseguir interesantes efectos. En el siguiente ejemplo, podremos escribir a un mismo tiempo un mensaje en horizontal y en vertical.

```
10 CLS
20 FOR N=0 TO 6
30 PRINT "MENSAJE ";SCREEN$ (0,N)
40 NEXT N
```

Este «mini programa», también se puede adaptar para escribir en alguna columna y fila distinta a la cero. Si llamamos F al número de fila y C al número de columna, podemos generalizar el programa dejándolo como:

```
10 FOR N=0 TO 6
20 PRINT AT F+N,C;"MENSAJE ";SCREEN$ (F+N,C+N)
30 NEXT N
```

El bucle debe comenzar desde cero y terminar en dos unidades menos que la longitud del mensaje. Por otra parte, el mensaje ha de terminar en un espacio en blanco, como separación del mensaje horizontal del vertical. Por tanto, podemos generalizar definitivamente la rutina, si suponemos que el mensaje a escribir se encuentra en la variable A$, con:

```
10 FOR N=0 TO LEN A$-2
20 PRINT AT F+N,C;A$;SCREEN$ (F+N,C+N)
30 NEXT N
```

BITS

Algunos compradores desconfiados, sobre todo de equipos de segunda mano, se preguntan como pueden comprobar que el Spectrum que van a comprar es efectivamente de 48 Kb. A continuación exponemos distintas maneras de cerciorarse de ello.

1) **PRINT PEEK 23733.** Si el modelo es de 48 Kb. debería aparecer el valor 255, de no ser así, 127.

2) **PRINT PEEK 23732+256*PEEK 23733.** El valor obtenido debería ser 65535, tratándose de un modelo de 48 Kb. y 32767, si fuera de 16 Kb.

Las direcciones de memoria investigadas en ambos casos, corresponden a la variable del sistema **P-RAMT** *(Phisical RAM top)*, que indica la dirección del último byte de la memoria física.

*

blanco, como separador del resto de la pantalla). Introduciendo un 0 en la variable **DF SZ (POKE 23659,0)**, cuando el ordenador intente mostrar cualquier mensaje de error, se producirá el denominado CRASH o BLOQUEO del sistema. El Spectrum intentará mostrar el informe en un lugar de la pantalla donde no tiene espacio. La única forma de acabar con esta situación será desenchufando el ordenador, o pulsando el botón de RESET, con la consiguiente pérdida de lo almacenado en la memoria.

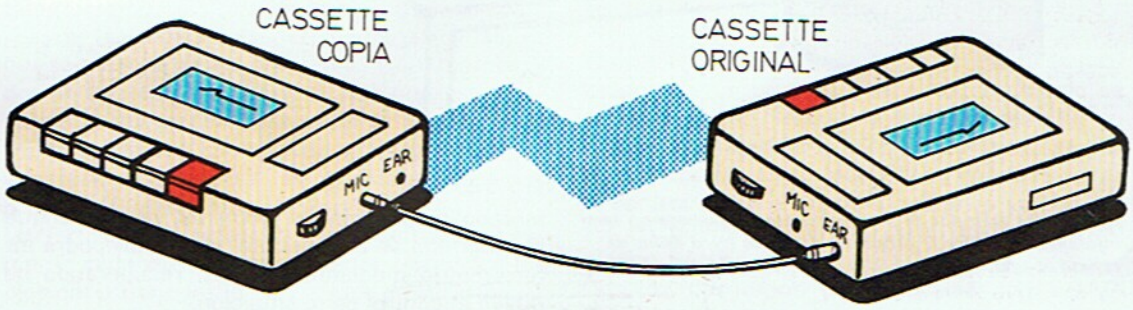
El inconveniente que lleva consigo la utilización de esta variable del sistema es que no podremos emplear en el programa el comando **INPUT**, debido a que éste realiza la toma de datos en la zona de pantalla afectada. Un posible método alternativo a **INPUT**, puede ser el uso de la función **INKEY$**.

Dentro del mismo estilo de la solución anterior, se encuentra el manejo de la variable del sistema **ERR SP**, que contiene la dirección a que debe acceder el Spectrum en caso de error. Si hacemos cero el valor de esta variable del sistema, **(POKE 23613,0)** y se produce alguna detención por error en el programa, el ordenador realizará la misma operación que cuando lo conectamos a la red, es decir, un **RANDOMIZE USR 0**; evitándose además el problema del INPUT.

Otra forma, quizás más elegante, de utilizar esta variable del sistema, es efectuando **POKE 23613,82.** Al asignarle este valor, se consigue que la tecla **BREAK** quede desactivada, o mejor dicho, que al ser pulsada, el ordenador salte a la misma línea donde fue interrumpido el programa. Esta sería la opción ideal, de no ser porque en los bucles **FOR-NEXT**, en los **GOTO**, y en los **GOSUB-RETURN** se modifica este valor 82, y sería necesario repetir el **POKE** cada vez que nos encontramos con estas instrucciones.

Con lo visto hasta el momento, hemos conseguido controlar la apertura de nuestros programas una vez que éstos se hayan ejecutado. Pero si alguien, en vez de cargar nuestro programa con **LOAD "nombre"**, lo hiciera con **MERGE "nombre"**, se produciría el mismo efecto que si al grabarlo no hubiésemos hecho uso de la autoejecución. En este caso, todos nuestros esfuerzos habrían resultado vanos.

Para obviar este inconveniente, disponemos de algunas soluciones «anti-merge». La primera consiste en grabar el programa como **BYTES** (y a ver quién es capaz de hacer un **MERGE** con un bloque de bytes). Para ello, debemos grabar no sólo la zona de memoria que contiene el programa BASIC, sino también el área comprendida hasta **STKEND.**



La protección de programas almacenados en casete nunca puede resistir la copia mediante dos magnetófonos de relativa calidad.

Por ejemplo, incluyendo las siguientes instrucciones al comienzo del programa:

```
10 LET A=PEEK 23635+256*PEEK 23636
20 LET B=PEEK 23653+256*PEEK 23654
30 SAVE "nombre"CODE A,B-A+1
40 REM Comienza nuestro programa.
```

Siguiendo el ejemplo anterior, cuando carguemos mediante **LOAD "nombre" CODE**, el programa se autoejecutará a partir de la línea 40; y cualquier intento de **MERGE** será rechazado por el sistema, al haber sido grabado el programa como **CODE** (bytes).

Los **PEEK** de 23653 y 23654, proporcionan la dirección de **STKEND**, mientras que los de 23635 y 23636, corresponden a la variable del sistema conocida como **PROG**, que indica el comienzo del área de texto BASIC. Por tanto, si cambiamos la dirección de comienzo de grabación del **CODE**, para que sea 16384, podremos almacenar también, a un mismo tiempo, la imagen que en el momehto de la grabación hubiera en la pantalla. Para ello, habría que sustituir la línea 10 del ejemplo anterior por **LET A=16384.**

En algunos programas quizás hayamos observado que la primera instrucción es una misteriosa línea 0, la cual no podemos editar ni borrar por los métodos normales, si bien sí listar y ejecutar. En esta línea, los programadores suelen introducir mensajes para anunciar la propiedad del programa o su autoría.

El método para introducir una línea 0 es bastante simple. En primer lugar, hemos de crear una primera línea de programa mediante el sistema habitual. Esta es la línea de instrucción a la que, más adelante, rectificaremos el número original para hacerlo 0. Por ejemplo:

1 REM (C) Enrique Fulano de Tal * Para la realización de este programa, empleé tres meses de trabajo. Por favor, respete mis derechos sobre él.

El siguiente paso es averiguar la dirección de memoria en que está almacenado nuestro programa. Cuando no tenemos conectado el ZX Interface 1, este punto inicial es siempre el mismo (23755). De no ser así, será necesario que pongamos en práctica el método que vimos unas líneas más arriba al estudiar la grabación de programas como CODE. Recordémoslo ahora:

PRINT PEEK 23635+256*PEEK 23636

Al hacer ésto, el ordenador imprimirá en la pantalla un número que corresponde con la primera dirección de programa. Para conseguir convertir la línea 1 antes introducida, en línea 0, sólo es necesario realizar una última operación:

POKE X,0: POKE X+1,0

Donde X, será sustituido en cada caso por la dirección de inicio del programa, es decir, la contenida en la variable del sistema **PROG (PEEK 23635+256*PEEK 23636),** que antes ha aparecido en la pantalla.

La técnica de la introducción de la línea 0 tiene una doble utilidad. Por una parte, como ya hemos visto, permite incluir en el programa BASIC un mensaje de *copyright* que no es suprimible por los cauces normales.

Por otra parte, es otro método «anti-MERGE». El comando **MERGE** realiza la fusión de programas y, por tanto, debe tener en cuenta los números de instrucción. De esta manera, cuando se encuentra un número de línea imposible, como es el caso de una línea cero, se produce un error que aboca al CRASH del sistema.

Lo cierto es que, sea el que sea el método de protección utilizado, cualquier usuario medianamente avanzado se encuentra en condiciones de poder acceder a él. Por ésto, todos nuestros esfuerzos han de ir encaminados, cuando menos, a dificultar la labor de «apertura» del programa y, como mínimo, garantizarle al «pirata» algún que otro dolor de cabeza.

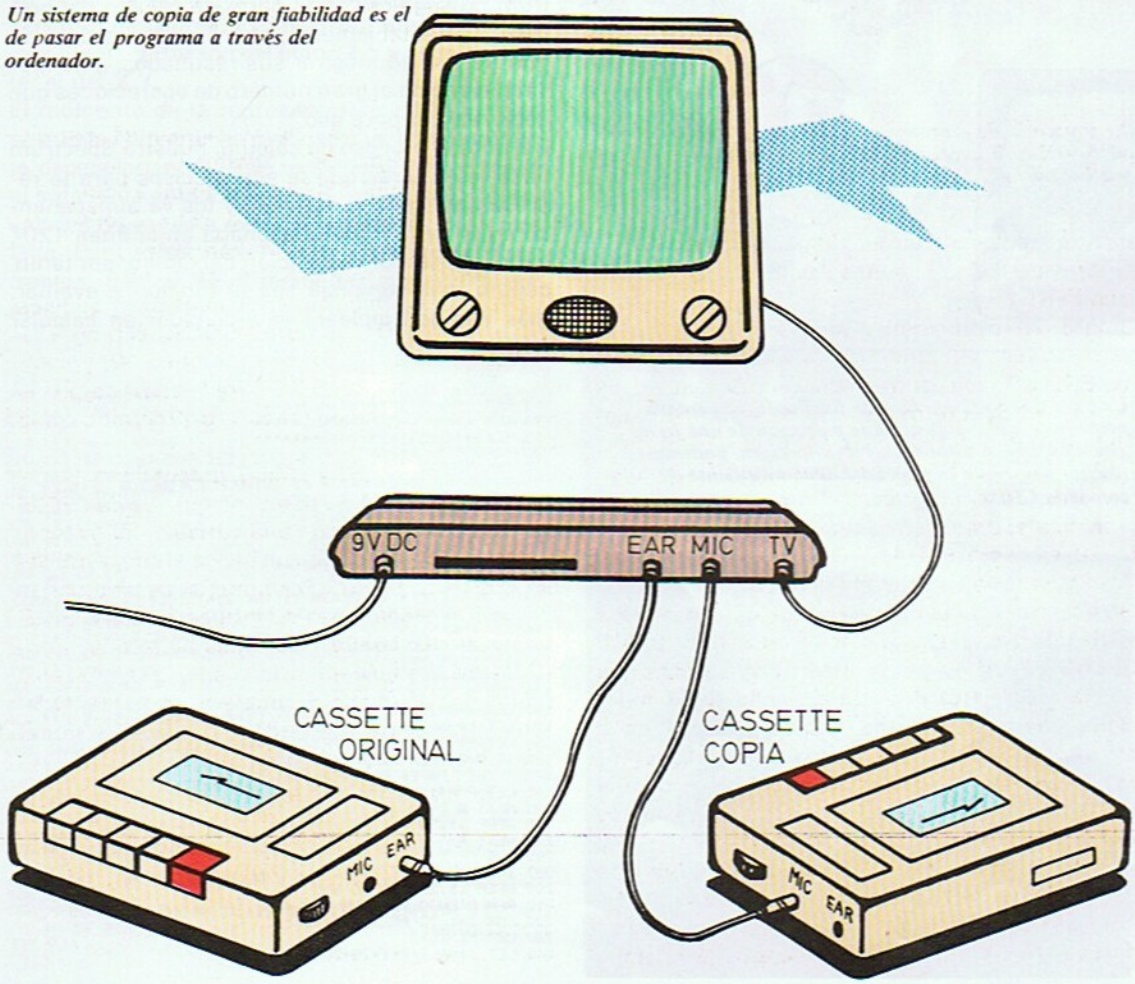
BITS

Una de las técnicas que más se utilizan en la impresión de literales en pantalla, es el centrado de los mismos. La rutina que aparece a continuación, centra un mensaje contenido en la variable A$, que será impreso en la fila F.

```
10 LET F=0: REM Linea a escribir
20 LET A$="MENSAJE": REM Mensaje a escribir
30 PRINT AT F,16-LEN A$/2;A$
```

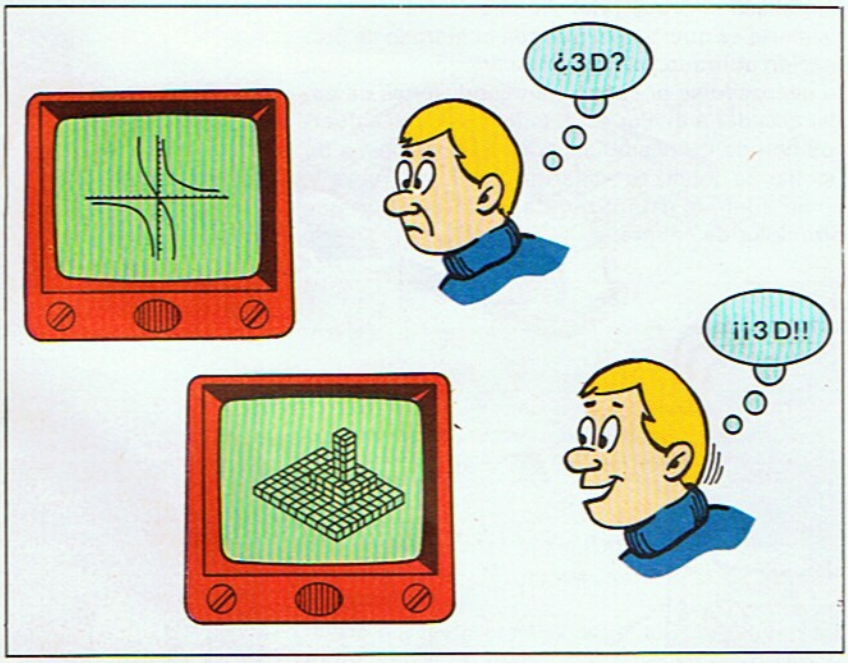
*

Un sistema de copia de gran fiabilidad es el de pasar el programa a través del ordenador.



FUNCIONES 3D

El programa de esta semana hará las delicias de todos aquellos que disfrutamos viendo como nuestro Spectrum hace evolucionar en la pantalla su invisible «lápiz mágico», ejecutando auténticas obras de arte cibernético.



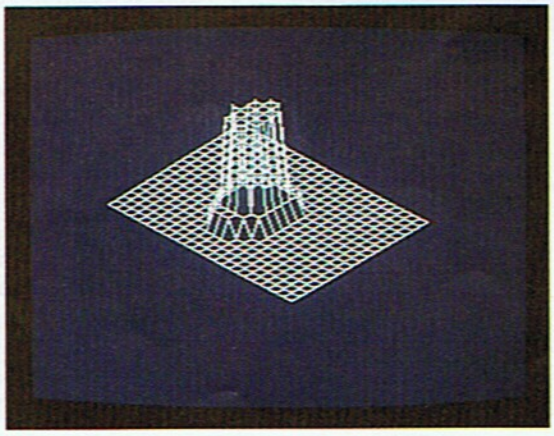
El programa en cuestión hace que nuestro Spectrum dibuje funciones matemáticas, aunque de una manera un tanto peculiar: ¡en tres dimensiones!

Para conseguir tan impresionantes representaciones sólo es necesario indicar un dato al ordenador: la función a evaluar. Una vez hecho ésto, el resto del proceso será gestionado automáticamente por el aparato.

Al ejecutar el programa, éste nos pide que introduzcamos la función que deseamos representar. Para ello, utiliza un **INPUT** con el título "F (X,Z)", con el cual nos quiere dar a entender que la ecuación a representar debe estar en función de los parámetros variables **X** y **Z**. De no hacerlo así, el programa cometerá, irremediablemente, un error. Una vez introducida la función deseada, el Spectrum proseguirá con la parte de cálculo del proceso. Lamentablemente, el ordenador tardará algún tiempo en mostrar sus resultados; algo inevitable debido al gran número de operaciones que debe realizar para ello.

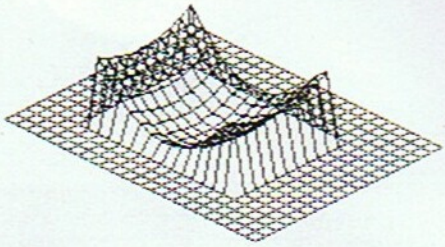
Durante el proceso de cálculo, nuestro Spectrum halla los valores que le son precisos para la representación de la función, y los va almacenando en la tabla P (dimensionada en la línea 120), para la posterior realización del dibujo; por tanto, cuanto más compleja sea la función a evaluar, más tiempo empleará el ordenador en calcular

Una vez que el dibujo ha sido finalizado, la pantalla permanece inalterable hasta la pulsación de una tecla; gracias a ello, si disponemos de impresora, podremos realizar una copia en papel del dibujo mediante el comando COPY.

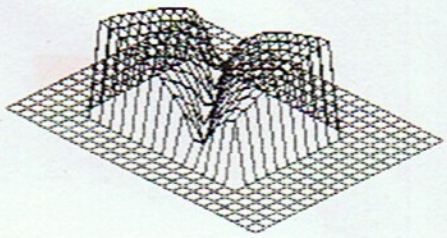


```
  5 REM ************************
 10 REM ***** J.M.MAYORAL *****
 15 REM ************************
 20 BORDER 0: PAPER 0: CLS
 30 INK 9
 35 PRINT AT 10,4; PAPER 6;'INTRODUCE FUNCION  F(X,Z)'
 40 LET LRG=256: LET ALT=176:
 50 LET XG=5: LET ZG=XG-2
 60 LET CT=INT (LRG/XG/2)
 70 LET PF=INT (ALT/ZG/3)
 75 INPUT 'F(X,Z)= '; LINE A$
 80 CLS
 90 PRINT  PAPER 6; FLASH 1; INK 2;AT 10,9;'  CALCULANDO
 95 PRINT AT 20,5;'   ESPERA SENTADO  ! '
 96 LET P=0
100 REM CALCULO GRAFICO
120 DIM P(CT,PF)
130 FOR A=-PF/2 TO PF/2
140 FOR B=-CT/2 TO CT/2
150 LET X=A*20/CT: LET Z=B*20/PF
170 LET Y=VAL A$
180 LET P((B+CT)/2,(A+PF)/2)=Y*ALT*(-1)
181 IF P=8 THEN  LET P=0
183 PRINT  PAPER P; OVER 1;AT 20,5;'
185 LET P=P+1
190 NEXT B
195 BEEP .1,30
200 NEXT A
205 CLS
207 INK 7
209 BEEP .1,20: GO TO 350
210 REM DIBUJO PLANO X-Y
220 FOR Z=1 TO PF
230 LET X1=XG*Z
240 LET Z1=ALT/2+Z*ZG+20*(-1)
```

su representación, aunque, por otra parte, es con este tipo de funciones con las que se suelen obtener unos gráficos más espectaculares.



(X*X+Z*Z)/1000



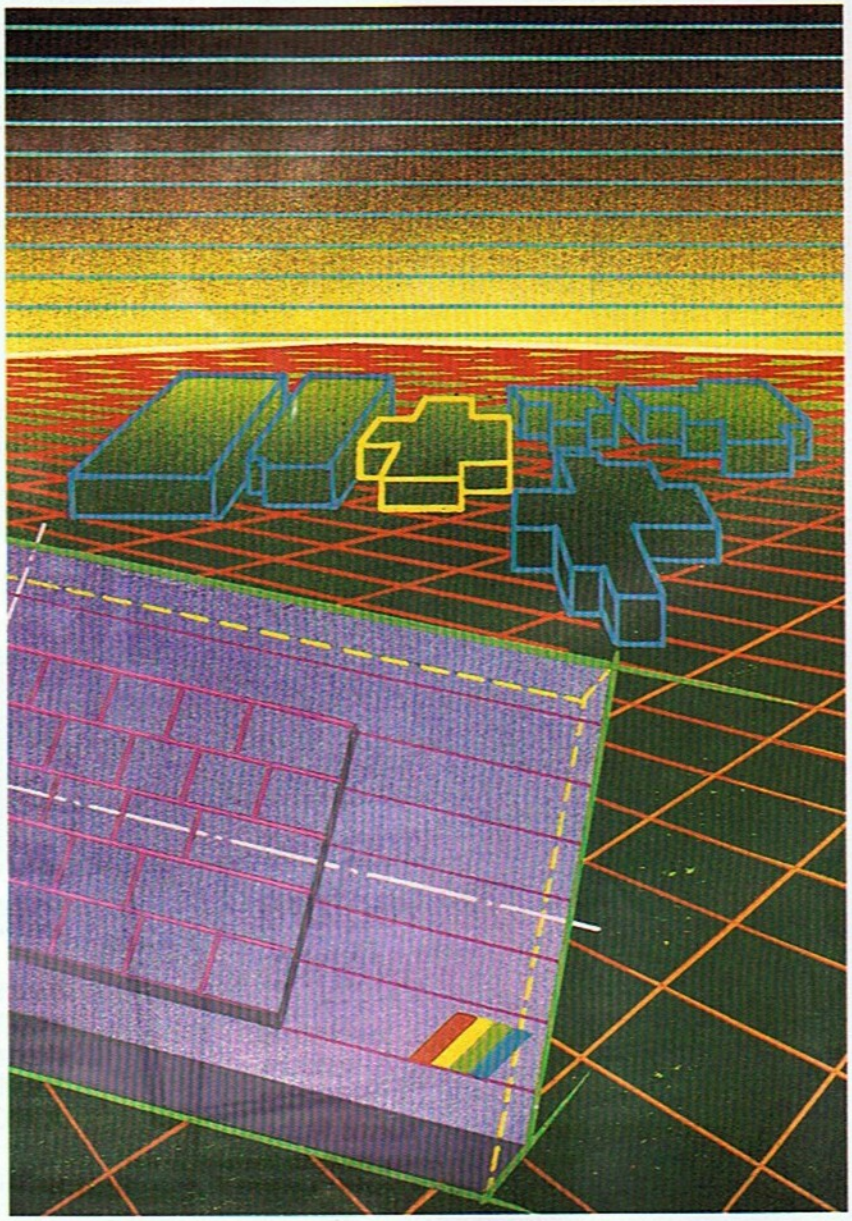
LN (1+(X+Z)*SGN (X+Z))/10

REPRESENTACION DE LA FUNCION

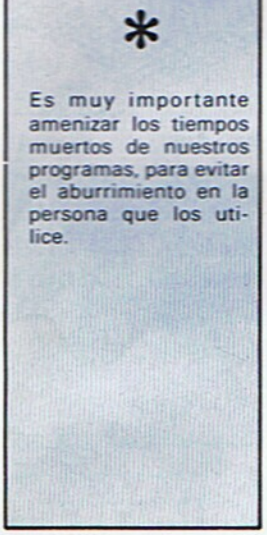
¡Bien! La espera llegó a su fin. Una vez terminado el proceso de cálculo, comenzarán a aparecer en la pantalla las líneas que configuran la imagen tridimensional. El Spectrum efectúa en primer lugar, el trazado del plano Z-Y y seguidamente, el del X-Y.

El momento de la representación de la función, se puede convertir en glorioso o en absolutamente decepcionante, sobre todo si no hemos tenido el suficiente cuidado de elegir una función cuyos valores no vayan a exceder los límites de la pantalla. Para evitar este tipo de experiencias frustrantes, hemos de observar las siguientes normas:

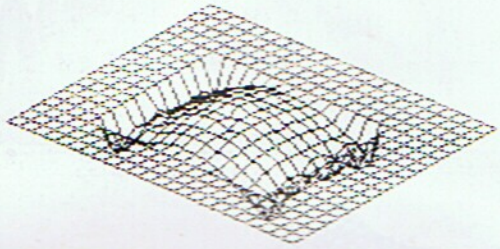
```
250 LET XN=X1+XG
260 LET ZN=Z1-ZG-P(1,Z)
270 FOR X=1 TO CT
280 LET XP=X1+X*XG
290 LET ZP=Z1-X*ZG-P(X,Z)
300 PLOT XN+15,ZN+20
305 DRAW XP-XN,ZP-ZN
310 LET XN=XP
320 LET ZN=ZP
330 NEXT X
335 NEXT Z
337 GO TO 470
340 REM DIBUJO PLANO Z-Y
350 FOR X=1 TO CT
360 LET X1=XG*X+PF*XG
370 LET Z1=ALT/2-X*ZG+PF*ZG+20*(-1)
380 LET ZN=Z1-ZG-P(X,PF-1)
390 LET XN=X1-XG
400 FOR Z=0 TO PF-1
410 LET XP=X1-Z*XG
420 LET ZP=Z1-Z*ZG-P(X,PF-Z)
430 PLOT XN+15,ZN+20
435 DRAW XP-XN,ZP-ZN
440 LET XN=XP
445 LET ZN=ZP
450 NEXT Z
460 NEXT X
465 GO TO 220
470 REM VISUALIZACION
480 FOR F=0 TO 21
490 PRINT  INK 5; OVER 1;AT F,0;"
500 NEXT F
700 PRINT AT 21,1;"PULSA UNA TECLA PARA EMPEZAR"
710 PAUSE 0
720 RUN
```



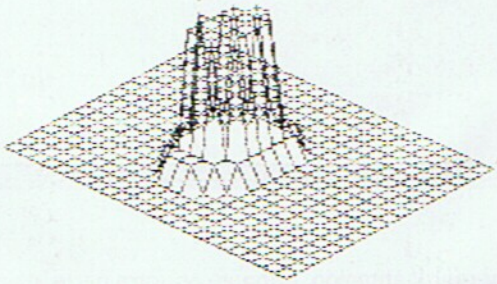
La grabación del programa puede efectuarse mediante SAVE "GRAF. 3D" LINE 10.



Es muy importante amenizar los tiempos muertos de nuestros programas, para evitar el aburrimiento en la persona que los utilice.



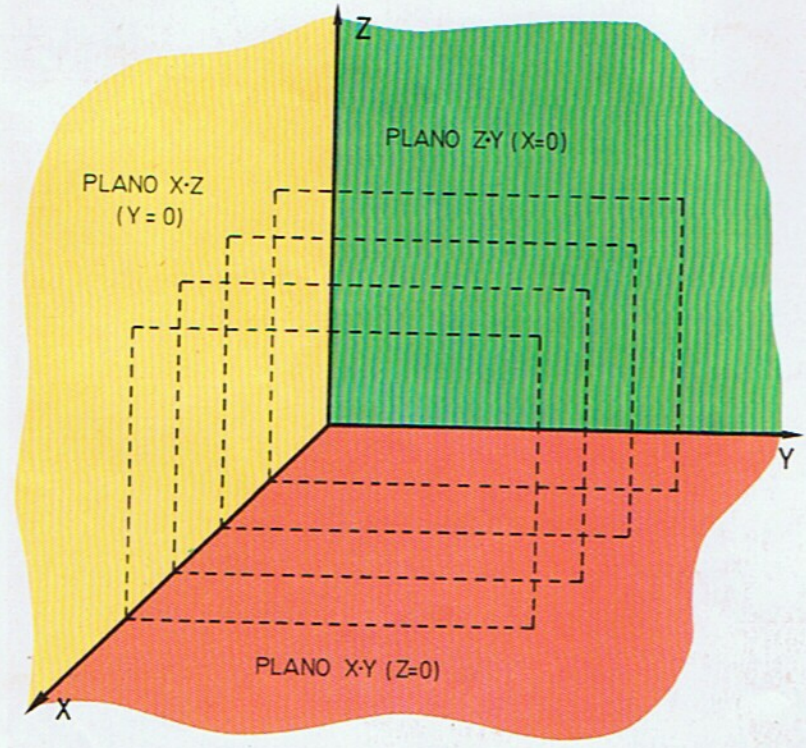
```
SIN (((60-X*X-Z*Z)*(ABS SIN (60-
X*X-Z*Z)+1))/3000)
```



```
SGN (INT (23/(.00001+X*X+Z*Z)))/
3+SGN (INT (55/(.00001+X*X+Z*Z))
)/15
```

La realización de dibujos en tres dimensiones es un efecto de gran espectacularidad en los programas.

- Evitar el empleo de cocientes con denominador variable, ya que si éste se hace cero, la función tendrá un valor indeterminado en ese punto. Por ejemplo, X/Z.
- Evitar operaciones matemáticas no contempladas por el Spectrum, como es el caso de las raíces de radicando negativo y las potencias de base negativa. Por ejemplo, (X-Z)↑2 o SQR (X-Z).
- Utilizar la función BASIC **SGN** para evitar el caso anteriormente expuesto. Por ejemplo, SQR ((X-Z) * SGN (X-Z)).
- Sustituir en lo posible las potenciaciones por las correspondientes series de multiplicaciones, para evitar los errores de bases negativas. Por ejemplo, emplear X*X+Z*Z, en vez de X↑2 + Z↑2.



Para la representación de las funciones en tres dimensiones, se emplea un sistema de proyección en los planos Z-Y y X-Y.

- Evitar los valores infinitos de las tangentes. Por ejemplo, TAN (X + Z).

Si tenemos cuidado con todo ésto, lo más probable es que tras el tiempo de espera en el diseño del gráfico, se colmen nuestras aspiraciones con una representación más o menos afortunada. En todo caso, siempre podremos empezar por comprobar el correcto funcionamiento del programa, introduciendo las funciones de ejemplo que aparecen en estas páginas.

A diferencia del programa de nuestro pasado número, la introducción del que ahora nos ocupa no nos supondrá un gran esfuerzo, debido a su escasa longitud. Tampoco tendremos el problema de los caracteres subrayados, puesto que en su realización no se han empleado gráficos definidos.

Por tanto, para la adopción del programa no hay que seguir ningún método especial y será suficiente, una vez introducido el listado, grabarlo mediante el pertinente comando **SAVE.**

Para finalizar, fijémonos en cómo se han tratado los tiempos de espera en el programa. Durante el período más o menos prolongado de cálculo de los valores de la función aparecen en la pantalla dos rótulos, que debido a su dinamismo (cambian constantemente de color) hacen ligeramente más soportable el tiempo muerto.

El consejo de amenizar los tiempos de espera con este tipo de mensajes, no es algo que debamos dejar caer en saco roto. Para comprobar la efectividad de esta estratagema, no tenemos más que suprimir las líneas 90 y 95 del programa. De esta manera observaremos como el tiempo se dilata sensiblemente, aunque sólo de manera aparente, pudiendo llegar a hacer dudar del correcto funcionamiento del programa.

Con la misma intención que los mensajes dinámicos, se ha introducido una instrucción BEEP, que emite un sonido por cada paso del cálculo. Este último detalle, también contribuye en gran manera a amenizar la espera en los tiempos muertos.

UNA SUPERCALCULADORA

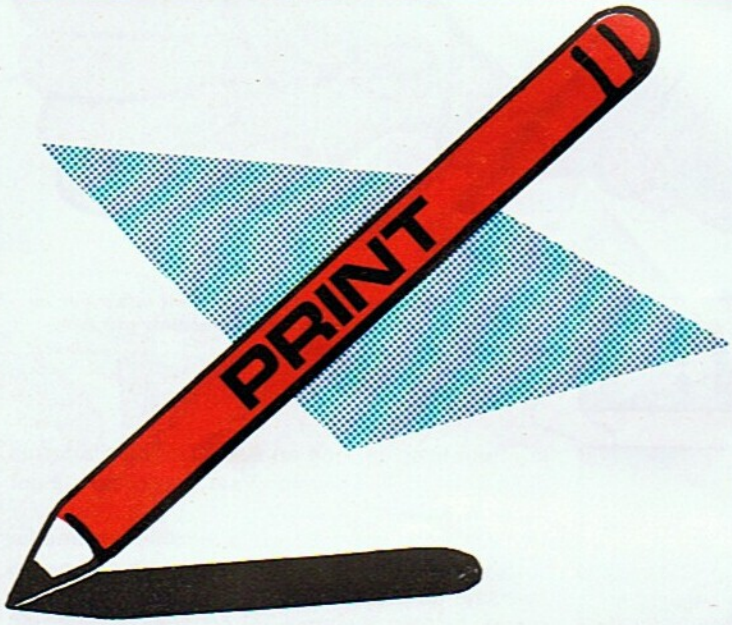
POR fin ha llegado el momento de que empecemos a sacar provecho de los conocimientos sobre el teclado que hemos acumulado en los capítulos anteriores. A partir de ahora, vamos a trabajar en lo que en el BASIC se llama MODO DIRECTO, que nos va a permitir obtener del ordenador los resultados de operaciones matemáticas propuestas por nosotros mismos.

Para comenzar con buen pie en esta nueva fase, debemos conocer primero una de las más útiles y populares instrucciones del repertorio BASIC: la instrucción **PRINT.**

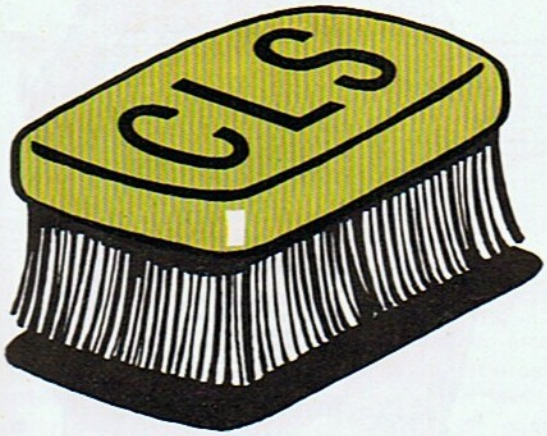
PRINT permite la representación en pantalla de números, letras y símbolos, así como resultados de las operaciones ejecutadas por el ordenador. Del mismo modo que nosotros comunicamos nuestros deseos al Spectrum a través del teclado, éste nos contesta con resultados impresos sobre la pantalla por medio de la instrucción **PRINT.** Es la forma de comunicación visual que el ordenador dispone, para que podamos interpretar las informaciones que nos facilita.

Debemos saber también que las órdenes que introducimos en el Spectrum a través de la última línea de la pantalla, deben terminar siempre con la pulsación de la tecla **ENTER,** que comunica al ordenador que hemos concluido la introducción de los datos. Esto es bastante lógico, porque de no ser así, el ordenador nunca sabría cuando habíamos terminado de introducir la orden completa, ni por tanto cuando comenzar a ejecutarla. Vamos a ver también otra instrucción, **CLS,** que nos va a permitir borrar el contenido de la pantalla, para no mezclar los nuevos resultados obtenidos con los presentes ya en la pantalla procedentes de operaciones anteriores.

Una vez hechas estas breves descripciones de las instrucciones que emplearemos, podemos comenzar con el comentario de las operaciones aritméticas básicas.



La sentencia PRINT *es la herramienta BASIC que utilizamos para escribir en la pantalla del Spectrum.*



A la hora de borrar una pantalla, el sistema más eficaz es la sentencia CLS.

ARITMETICA EN NUESTRO SPECTRUM

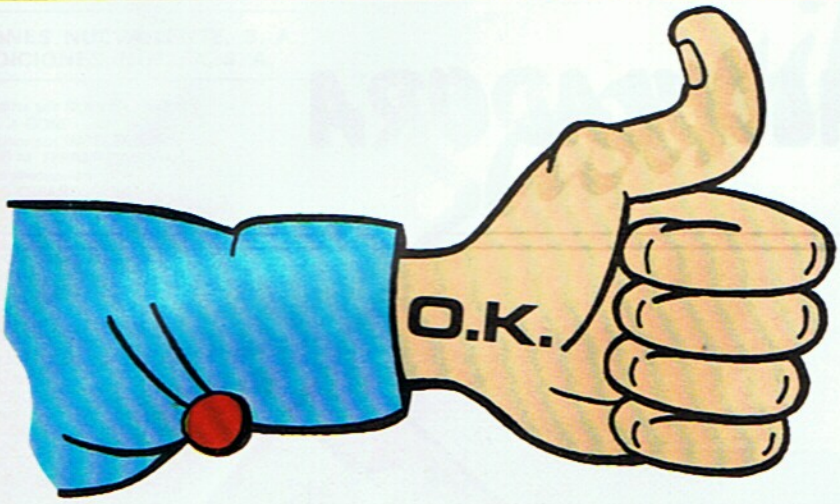
Las operaciones de suma, resta, multiplicación y división, se concretan en BASIC a través de los símbolos +, –, * y /, respectivamente. Acostumbrados a los símbolos aritméticos tradicionales, nos encontramos con las únicas innovaciones de

¡!

La realización de las operaciones aritméticas básicas, se lleva a cabo en el ordenador, mediante el uso de los símbolos +, –, * y /, para las operaciones suma, resta, multiplicación y división, respectivamente.

*

Recordaremos que sólo cuando dos operaciones tienen prioridades iguales, se resuelven de izquierda a derecha.



Mediante el mensaje OK, *el Spectrum nos indica que ha concluido sin problemas su último trabajo y que está dispuesto para el próximo.*

¡!

Son nombres válidos para variables numéricas, cualquiera que comience por una letra y sólo esté constituido por letras y/o números. No se admitirá ningún símbolo en su composición, aunque sí espacios en blanco. Tampoco existe restricción en cuanto a la longitud del nombre.

la multiplicación, que en BASIC adopta la forma del asterisco, y la división, que se representa por la barra inclinada o *slash*.

Comencemos por conectar a la red el ordenador y, después de una breve pausa hasta obtener el mensaje de presentación, pulsemos las siguientes teclas: **P**, que nos muestra la instrucción **PRINT**; la tecla del **1**, con el cursor ya en el modo **L**; a continuación el símbolo **+**, que obtenemos por pulsación de **SYMBOL SHIFT** y **K**; y por último nuevamente la tecla **1**.

Si todo ha marchado correctamente, debemos encontrar en la última línea de la pantalla exactamente ésto:

PRINT 1+1

Con el cursor situado inmediatamente a la derecha del último 1 escrito, y en el modo **L**. Ya sólo falta que pulsemos la tecla **ENTER**, para que el Spectrum comprenda que hemos concluido la introducción de la instrucción y nos muestre el resultado de la operación solicitada. Hecho ésto, aparece en la línea superior de la pantalla un **2**, y en la última línea de la pantalla, donde estaba situada nuestra orden, el mensaje:

0 OK, 0:1

Más adelante, entraremos en profundidad en el análisis de este mensaje. De momento debe bastarnos con saber que el *OK* nos dice que todo ha marchado correctamente, y no se han producido errores.

Es imprescindible que nos demos cuenta ahora, de la importancia y la utilidad de la tecla **ENTER**. Durante toda la introducción de esta instrucción, que ha ordenado a nuestro Spectrum que calcule el resultado de la suma de 1 + 1 y nos la muestre en la pantalla, el cursor ha estado siempre presente para indicarnos qué clase de carácter esperaba a continuación y, al final, se ha quedado parpadeando mostrando la letra **L**.

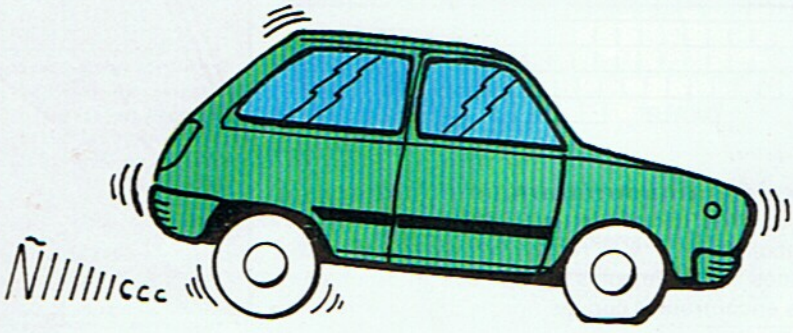
Si nosotros no pulsamos la tecla **ENTER**, el ordenador no puede «adivinar» que hemos terminado la instrucción; es más, si detectamos un error de sintaxis antes de pulsar **ENTER**, estamos a tiempo de corregirlo volviendo hacia atrás con la tecla de cursor a la izquierda (**SHIFT** y **5**) y borrando el carácter erróneo con **DELETE** (**SHIFT** y **0**), para reemplazarlo a continuación por el correcto, antes de pulsar definitivamente **ENTER**.

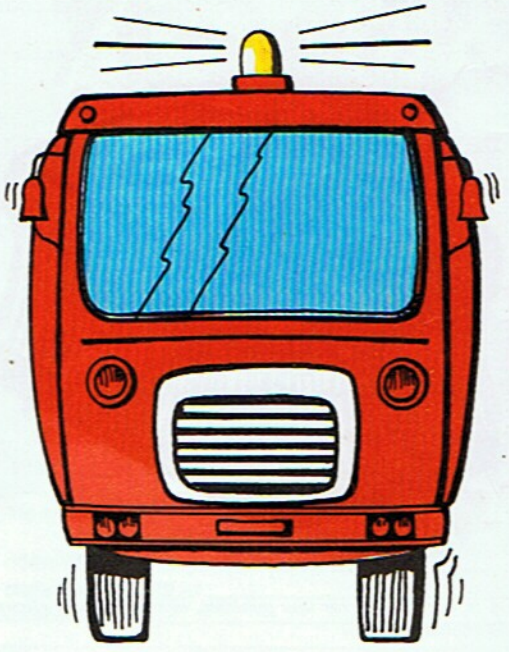
Para experimentar con el resto de las operaciones aritméticas básicas, podemos proponer un ejemplo algo más complicado que las conjugue todas:

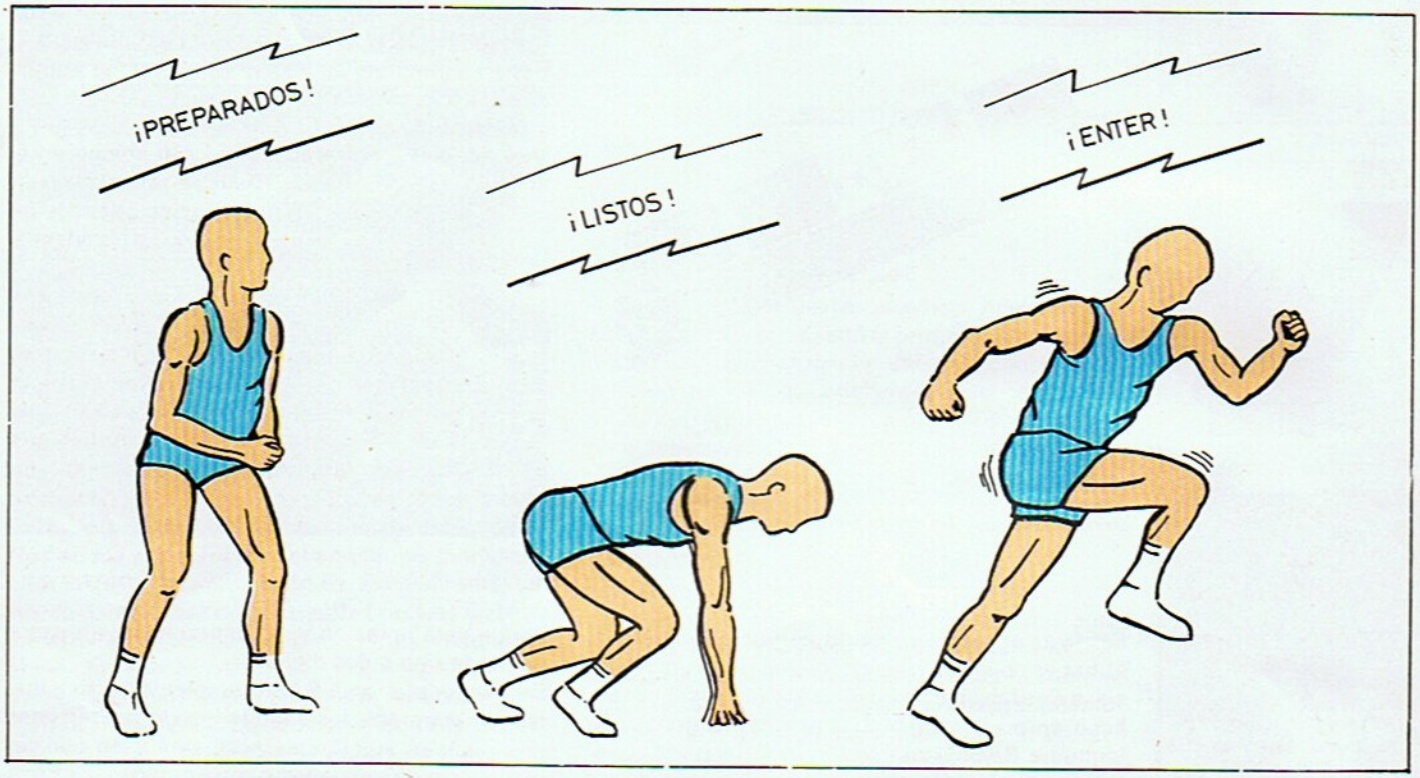
PRINT 5-2+3*2/3

Después de introducida la instrucción y pulsando **ENTER**, obtenemos el resultado **5**, que se impri-

Al igual que en la circulación existen prioridades de paso, el Spectrum sigue el mismo sistema para la realización de operaciones matemáticas combinadas.







mirá en la pantalla justo debajo del obtenido en la operación de suma anterior.
Del mismo modo, en la línea en que introducimos nuestras instrucciones, aparecerá el mensaje:

0 OK, 0:1

Llegando a este punto, lo lógico es que nos encontremos un tanto desconcertados por el mensaje que el Spectrum nos muestra en esta ocasión, porque, al menos desde nuestro punto de vista, la operación no ha sido ejecutada correctamente. Es evidente que el resultado esperado, aplicando nuestra forma tradicional de operar, de izquierda a derecha, habría sido **4**.

PRIORIDADES EN EL CALCULO

Es el momento de analizar a qué es debido este estado de cosas. Lo que sucede es que el ordenador dispone de una organización interna de cálculo, que le impone prioridad a la hora de realizar unas operaciones sobre otras. Cuando las prioridades le resultan iguales, opera de izquierda a derecha, de la misma manera que lo hacemos nosotros.

No olvidemos pulsar ENTER *cuando terminemos de escribir una instrucción.*

Centrándonos en el ejemplo anterior, primero se calcula el resultado de 3*2, después se divide éste por 3, calculando el resultado intermedio 2; por último se calcula el resultado intermedio de 5–2, 3, sumándose definitivamente los dos resultados intermedios anteriores para obtener: 5. De lo dicho, se desprende que la multiplicación y la división tienen igual prioridad entre sí, pero superior a la de la suma y la resta. A su vez, estas dos últimas operaciones tienen prioridades ideńticas.
Otras operaciones, como la potenciación y las funciones, tienen mayor prioridad que la multiplicación y división, y otras como el signo – que precede a un número para indicar que éste es negativo, tienen prioridad sobre todas las anteriores. Realmente, existe una complicada y estricta jerarquización de las operaciones (cuadro), pero nosotros tenemos oportunidad de alterarla a nuestro gusto empleando los paréntesis. Cualquier operación encerrada entre paréntesis, tiene prioridad absoluta sobre las demaś.
Con estos conocimientos, podemos ahora escribir la operación anterior forzando que el orden de cálculo se efectúe rigurosamente de izquierda a derecha, de esta forma:

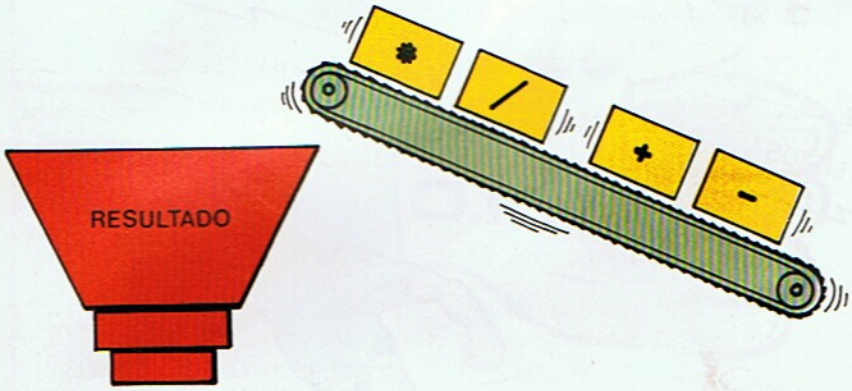
PRINT (5-2+3)*2/3

¡!

Las operaciones no son realizadas por el Spectrum de la misma forma que lo haríamos nosotros, es decir, de izquierda a derecha. El orden en que se resuelven viene dado por una determinada prioridad.

*

Llamamos números naturales, a los números positivos no decimales (1, 2, 3, 4, 5, 6, 7...).



Las operaciones multiplicación y división, se realizan siempre antes que las sumas y restas.

PRIORIDAD DE LAS OPERACIONES

Operación	*Prioridad*
Asignación de subíndices y fragmentación	12
Todas las funciones salvo **NOT** y operación unaria "menos"	11
Potenciación	10
Operación unaria "menos"	9
Multiplicación y división	8
=, >, <, <=, >=, <>	6
NOT	5
AND	4
OR	3
	2

Esta vez el resultado de la operación sí será **4**, al haber obligado, con el empleo de los paréntesis, a que las operaciones de suma y resta se realicen en primer lugar. Los paréntesis dentro del lenguaje BASIC, son la forma de especificar al Spectrum el orden concreto de ejecución de las operaciones.

Hasta ahora hemos empleado números naturales en las operaciones. Podemos también experimentar con números negativos, reales y en notación científica.

Para introducir números negativos, basta con precederlos por el signo –. Del mismo modo, para números decimales, podemos incluir el punto decimal, como se hace en una calculadora convencional.

También, al igual que en algunas calculadoras, podemos emplear la notación científica. Este sistema consiste en introducir primero una mantisa con o sin punto decimal, la letra **E** como indicador de esta notación y, a continuación, un exponente de uno o dos dígitos.

De hecho, el propio BASIC emplea este sistema, para mostrarnos los resultados de operaciones que rebasan sus límites normales, bien por ser números muy pequeños o muy grandes.

Del mismo modo, otra de las características del BASIC, es la de redondear de forma automática los resultados de las operaciones, que exceden en número de decimales su capacidad de cálculo. Podemos probar con dos ejemplos para ver mejor este hecho:

```
PRINT 10/3
PRINT 20/3
```

Los resultados serán, respectivamente, 3.3333333 y 6.6666667. Como podemos ver, el

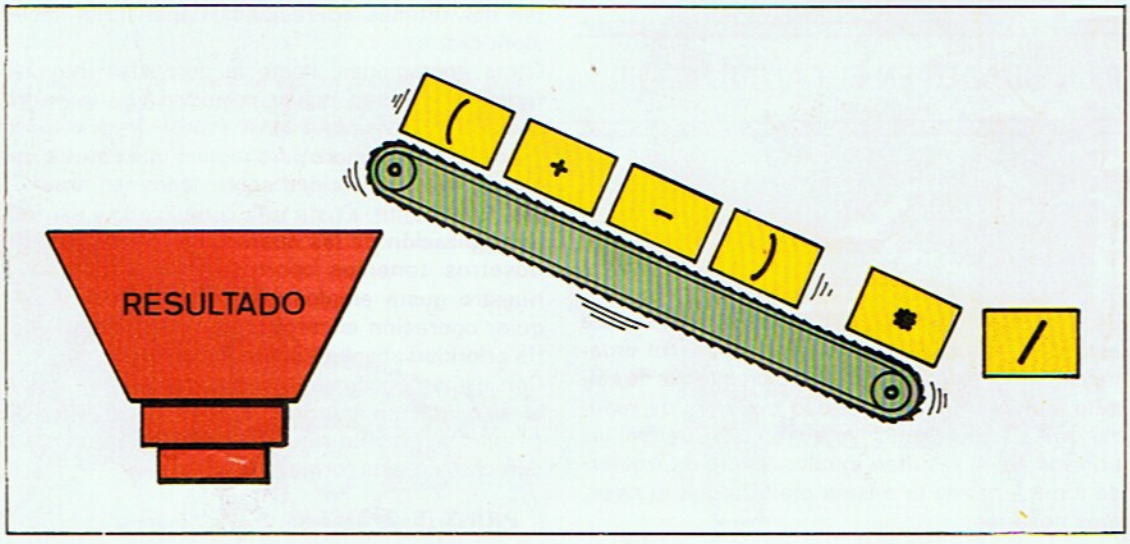
Ante los paréntesis no hay prioridad que valga.

¡!

El que sus nombres aparezcan en minúsculas, mayúsculas o combinación de ambos, no supone distinción alguna.

*

La sentencia BASIC para asignación de variables es **LET**, cuyo significado en castellano es "haz". Su sintaxis requiere que **LET** vaya seguido por el nombre de la variable a asignar, un símbolo de igualdad (=) y el valor de la variable. Por ejemplo: **LET A=3** (haz A igual a tres).



BASIC ha redondeado el resultado de la segunda operación para mostrarnos la respuesta más próxima a la verdadera, desde luego, dentro de sus posibilidades de cálculo.

Un ejemplo con el que podemos apreciar el cambio automático de notación, es pedir a nuestro Spectrum que nos represente en la pantalla un número muy grande. Este puede ser el caso:

PRINT 100000000

En pantalla nos aparecerá la respuesta 1E+8, que es la representación en notación científica de cien millones.

Es esencial que practiquemos mucho con las operaciones aritméticas, ya que las emplearemos muy frecuentemente cuando hagamos programas. Lo mejor que podemos hacer, es plantearnos unas cuantas operaciones y pedirle a nuestro Spectrum que las realice, para comprobar la forma exacta en que éste las lleva a cabo.

Nos queda por comentar el empleo de la instrucción BASIC **CLS.** Su misión es, simplemente, borrar la pantalla para que nuestras nuevas instrucciones **PRINT** no se mezclen con los resultados anteriores.

Vamos a comenzar por emplearla en el siguiente ejemplo. Partiendo del mensaje de **OK,** tecleamos **V** y **ENTER.**

Inmediatamente la pantalla aparece en blanco, y en la línea inferior el mensaje que ya nos es familiar.

La notación científica, nos sirve para "comprimir" determinados números, que por su longitud, sería demasiado lioso escribir en su forma convencional.

LOS LITERALES o Cadenas de caracteres "....."

A partir de ahora, las respuestas a nuestras preguntas aparecerán desde la primera línea de pantalla. Para familiarizarnos del todo con lo dicho sobre el modo directo, debemos practicar bastante con los conocimientos adquiridos, pero también debemos saber como complementar los resultados numéricos de las operaciones con literales alfabéticos. Lo vamos a ver a través de unos ejemplos.

PRINT "2+2=";2+2

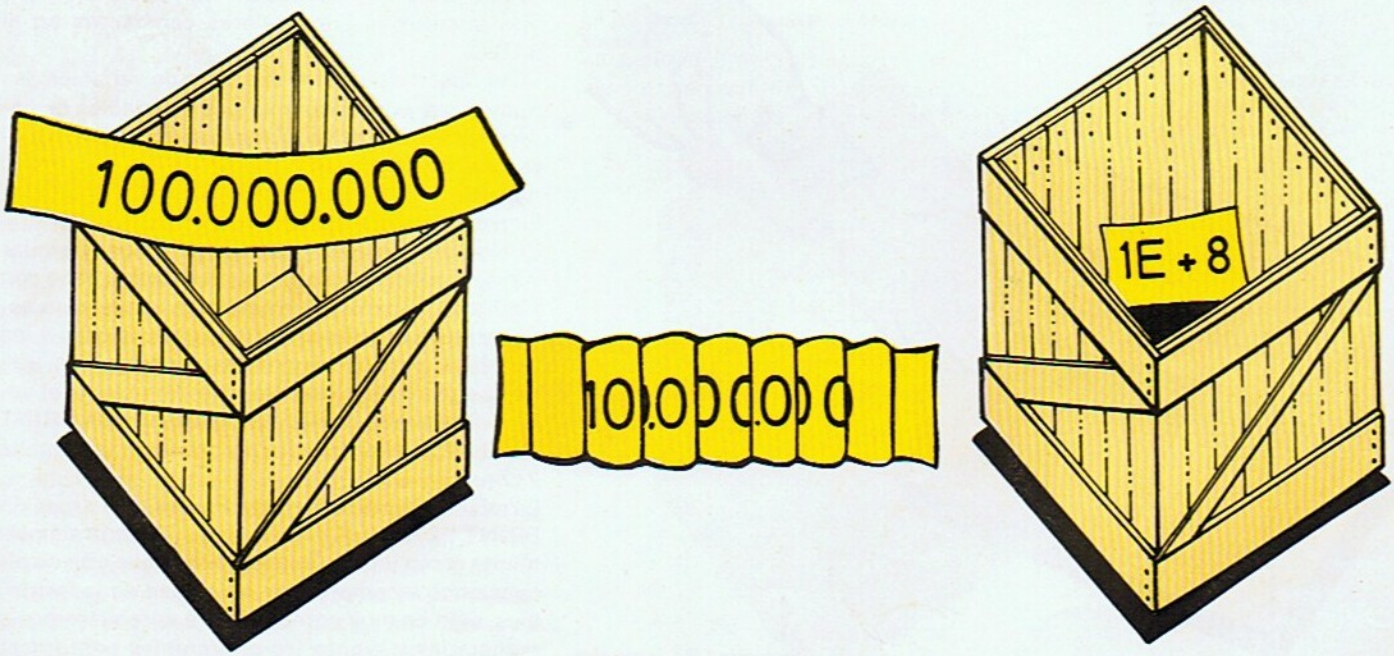
Los literales, también llamados cadenas de caracteres o *strings,* se introducen siempre entre comillas y son trasladados a la pantalla exactamente de la misma forma en que los escribimos (sin permanecer entrecomillados, por supuesto). El carácter ";" sirve de separador entre el literal y la fórmula situada a su derecha, que el ordenador debe resolver para obtener la respuesta. Los literales no tienen por qué emplearse conjuntamente con fórmulas, sino que pueden escribirse en la pantalla como simples frases. Por ejemplo, **PRINT "El Spectrum es un microordenador".** De lo visto hasta ahora se desprende, que la instrucción **PRINT,** sirve para escribir cualquier cosa en la pantalla: frases, resultados o combinaciones de ambos.

¡!

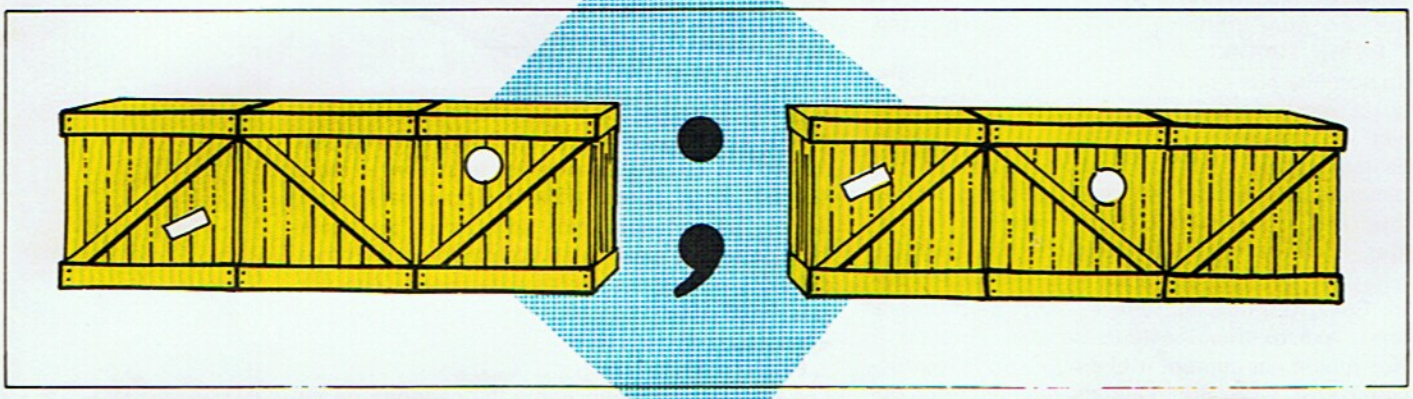
Son nombres válidos para variables de cadena, los que comienzan por una letra y tienen por segundo carácter un símbolo dólar ($). Su longitud máxima es dos (letra y dólar).

*

Las variables en BASIC pueden ser de dos tipos:
— Numéricas y
— de Cadena *(strings).*



Cualquier conjunto de caracteres encerrado entre comillas se considera un literal, y por tanto es escrito en la pantalla LITERALMENTE, es decir, sin intentar operarlo. Por ejemplo, **PRINT "2+2"**, no hace aparecer un 4, sino simplemente 2+2.

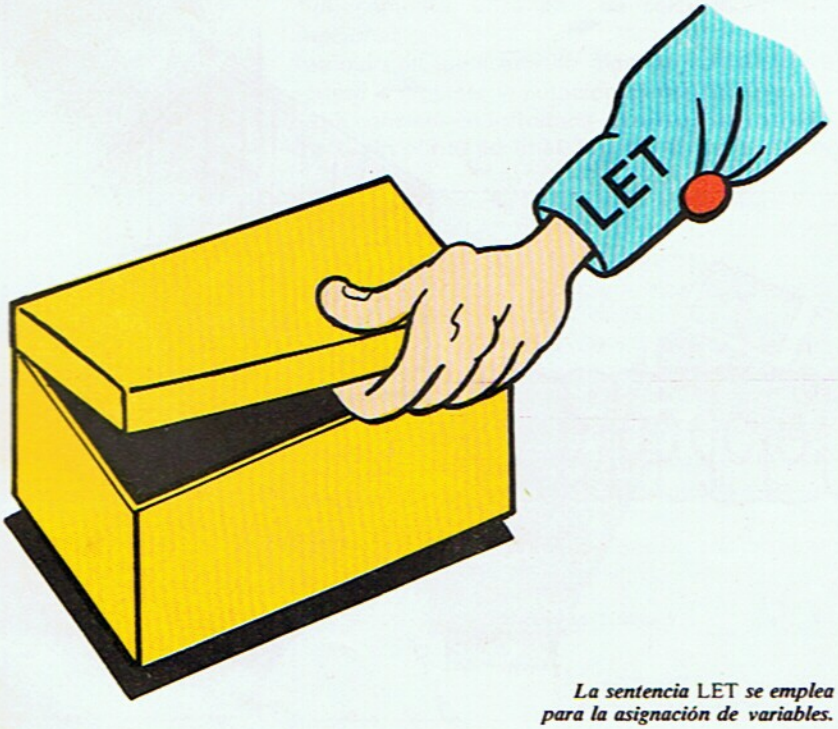


El punto y coma (;) actúa como separador entre las partes de una instrucción PRINT.

Como ya sabemos, también se pueden escribir resultados no encerrándolos entre comillas. Así por ejemplo, **PRINT 2+2**, proporcionará el resultado **4**. Finalmente, podemos combinar sin ningún problema literales y operaciones, separándolas mediante el carácter PUNTO Y COMA ;. Así por ejemplo, si ejecutamos **PRINT "2+2= ";2+2;"** y **3+7= ";3+7**, obtendremos:

2+2= 4 y 3+7= 10

El empleo de literales permite dar formato a las respuestas de los cálculos ejecutados por el ordenador. Más adelante, al comentar los primeros conocimientos sobre programación, veremos todo el partido que se puede sacar de las combinaciones de resultados calculados por el ordenador y los literales.



La sentencia LET *se emplea para la asignación de variables.*

LAS VARIABLES = Valores susceptibles de cambios.

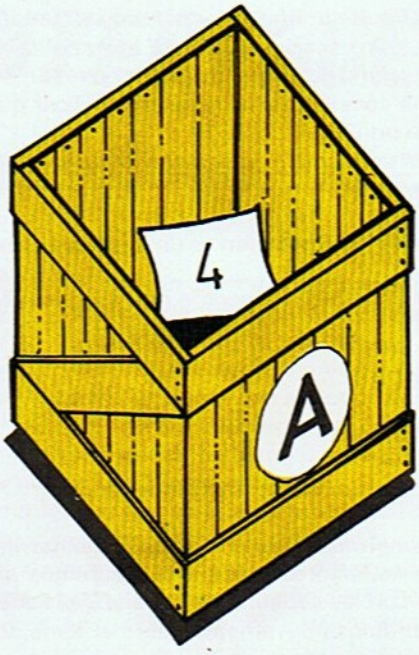
a) numéricas = contienen números
b) alfanuméricas = contienen litera

Hemos visto en el apartado anterior lo que son los literales y como imprimirlos en la pantalla. Pero, además de los literales que se escriben tal como son, podemos imprimir valores que son susceptibles de cambios. Estos valores se denominan VARIABLES. Al igual que los pronombres sustituyen a los nombres en las frases, las variables sustituyen a los valores constantes en el BASIC.

En el Spectrum existen dos tipos de variables: las numéricas y las alfanuméricas o variables de cadena *(strings)*. Las variables numéricas son capaces de contener cualquier número real, entero o natural, y las alfanuméricas cualquier literal. Empecemos hablando de las primeras.

Cualquier operación que pudiéramos efectuar con una constante, la podremos llevar a cabo con variables. Veamos un ejemplo: más adelante estudiaremos el método para asignar valores a las variables, pero por el momento imaginemos, que hemos otorgado a una variable llamada **X**, el valor **4**. Por tanto, al ejecutar la instrucción **PRINT "2+2= ";4**, el ordenador escribiría en la pantalla **2+2= 4**.

Lo mismo ocurriría si ejecutáramos la instrucción **PRINT "2+2= ";4**. Como vemos, **X** actúa simplemente como un sustituto de **4**. Aunque aún no alcancemos a comprender la utilidad de las variables, ésta es muy grande. Ahora aprenderemos a manejarlas y pronto las utilizaremos para hacer programas.



Las variables numéricas son similares a cajas, en las que el BASIC almacena valores numéricos.

LOS NOMBRES DE LAS VARIABLES

Las variables numéricas se denominan con un nombre de uno o varios caracteres, que debe comenzar necesariamente por una letra. Nombres válidos para variables numéricas son: G, Y1, RESULTADO, B1C, VALOR INICIAL, etc. Pueden emplearse, por lo tanto, combinaciones de letras y números, siempre que comiencen por una letra. Los nombres de variables numéricas pueden tener muchos caracteres de longitud e incluir blancos, gracias a lo cual podrán formar hasta frases completas. Hay que añadir que no se deben introducir símbolos en el nombre. En resumen, los nombres de las variables sólo pueden contener letras o números, y deben empezar por una letra. En lo referente al nombre, hay que decir que el BASIC no distingue en ese aspecto entre minúsculas y mayúsculas. Es decir, las variables SUMA, Suma, suma son la misma para el Spectrum.

Al finalizar el apartado anterior, imaginamos que indicábamos al ordenador el valor de la variable **X**. Ha llegado el momento de pasar de la imaginación a la acción. Más que a la acción, pasaremos a la ASIGNACION, ya que así es como se llama en informática al hecho de otorgar un valor a una variable. La instrucción BASIC para la asignación es **LET**, y debe ir seguida por el nombre de la variable a asignar, un símbolo igual (=) y el valor que vaya a tomar la variable.
Veamos algunos ejemplos de la asignación de variables:

```
LET CANTIDAD=50
LET PRECIO=25
PRINT "TOTAL= ";CANTIDAD*PRECIO
```

Hemos pedido a nuestro ordenador que memorice dos variables numéricas (CANTIDAD y PRECIO) y luego que las multiplique para hallar el total.
Los valores que hemos dado a las variables permanecen cautivos en la memoria del ordenador, de forma que podemos utilizarlos para repetir el cálculo, definiendo una nueva variable y empleando el resto de las anteriores:

```
LET TOTAL=CANTIDAD*PRECIO
PRINT "TOTAL= ";TOTAL
```

Como hemos podido ver, existen varias maneras de obtener los mismos resultados. Realmente, decidir la mejor forma de hacerlo depende de las necesidades particulares a la hora de programar.

En las variables de cadena, el ordenador almacena valores literales.

¡!

La utilización de paréntesis anula el orden de prioridades. De esta manera, cualquier operación entre paréntesis se resolverá antes que ninguna otra.

*

Número enteros son todos los positivos y negativos no decimales (... -3, -2, -1, 0, 1, 2, 3, 4...).

*

Las sentencias **PRINT** permiten la combinación de literales y números, ya sean constantes o en base a variables. Para este tipo de combinaciones, se utiliza el carácter; como separador entre literales y números.

*

La sentencia **LET** se utiliza tanto para la asignación de variables numéricas, como para literales. En este último caso, el valor de la variable a asignar deberá encerrarse entre comillas. Por ejemplo: **LET A$= "Hola"** (haz A$ igual a Hola).

¡!

El Spectrum puede actuar en todo momento como una calculadora, con gran cantidad de funciones matemáticas.

*

La sentencia **PRINT**, tiene la misión de escribir en la pantalla del Spectrum cualquier mensaje o resultado que nosotros queramos. *PRINT* es una voz inglesa, cuyo significado es escribir en letra de imprenta.

*

El borrado de la pantalla se consigue mediante la sentencia CLS, abreviatura de las palabras inglesas *Clear Screen* (borra pantalla).

*

No debe olvidarse nunca pulsar **ENTER** una vez que hayamos terminado de escribir la instrucción que deseamos introducir en el ordenador.

LAS VARIABLES DE CADENA

La misión de las variables de cadena es igual que la de las numéricas, es decir, sirven para sustituir una constante. Sin embargo, el tipo de constante a que equivalen, es evidentemente diferente. Mientras que las variables numéricas contienen valores numéricos, las de cadena sustituyen constantes literales.
Como es lógico, el BASIC necesita una manera de diferenciar las variables numéricas de las de cadena. Esto lo consigue mediante el nombre. Los nombres válidos para las variables de cadena están constituidos por una letra más el símbolo dólar (**$**). Dado que las minúsculas y las mayúsculas no son diferenciadas por el BASIC en los nombres de variables, existen únicamente 26 variables de cadena diferentes (**A$, B$, C$, D$,** ... **W$, X$, Y$** y **X$**).

Al igual que su diferenciación en el nombre, al utilizar la instrucción **LET** para asignar variables de cadena, no debemos olvidar encerrar entre comillas la constante a asignar para indicar que se trata de un literal.
Veamos un ejemplo de asignación de variable de cadena:

LET A$="El Spectrum es un microordenador"

Si ahora probamos a pedirle que escriba el valor de A$.

PRINT A$

Obtendremos...

El Spectrum es un microordenador

Del mismo modo que podíamos combinar literales con constantes numéricas, podremos ahora hacerlo con las variables de cadena. Más adelante incidiremos nuevamente sobre el tema de las variables; pero de momento, lo mejor que podemos hacer es practicar y practicar, hasta que consigamos dominar completamente el concepto de variables.

LA NOTACION CIENTIFICA

La notación científica se utiliza para representar números muy pequeños o muy grandes. Cualquier número representado de esta manera, está compuesto por cuatro partes fundamentales:

23 E + 7

La primera parte del número (23 en nuestro ejemplo) se denomina mantisa, y puede ser cualquier número real, es decir, decimal o entero y positivo o negativo.
A continuación se escribe una letra E, que sirve para indicar que el número se encuentra en notación científica (también conocida como "notación exponencial").
Inmediatamente después del indicador, aparece un signo positivo o negativo. En caso de que no figurara ningún signo, se supondría que se trata de uno positivo.
Por último, se escribe el exponente, que es un número natural con el que se determina la magnitud del número representado.
La forma de traducir un número en notación científica a notación convencional es muy fácil. Si el exponente es positivo (signo entre la E y el exponente), o bien no se ha indicado signo, el número será el resultado de multiplicar la mantisa por un uno seguido de tantos ceros como indique el exponente. Por ejemplo:

23 E + 7 = 23 × 10000000 = 230000000
-544 E 8 = -544 × 100000000 = -54400000000
6.463 E 10 = 6.463 × 10000000000 = = 64630000000

Por tanto, los exponentes positivos indican siempre números grandes, ya sean positivos o negativos. Por supuesto, cualquier número, aunque no sea muy grande, puede representarse en notación científica, aunque esto sólo complicaría el entendimiento de la cifra, en vez de facilitarlo. (Ej.: 2300 = 23 E 2 = 2.3 E 3).
Si el exponente es negativo, el número será el resultado de dividir la mantisa por un uno seguido de tantos ceros como indique el exponente. Por ejemplo.:

15 E -7 = 15/10000000 = 0.0000015
0.26 E -10 = 0.26/10000000000 = = 0.000000000026
-3.511 E -5 = -3.511/100000 = -0.00003511

Por tanto, los exponentes negativos se utilizan para representar números muy pequeños (muy próximos a cero). Como en el caso anterior, esta notación también puede emplearse para escribir números no muy pequeños, pero carecería de utilidad. (Ej.: 38 × E -3 = 0.038).

INTERFACES PARA JOYSTICKS

Ya tuvimos oportunidad anteriormente de tratar el tema de los *joysticks,* y quedó bien clara la necesidad de acoplar un interface a nuestro Spectrum para poder funcionar con este tipo de mandos. Vimos entonces cuáles eran las características más importantes de estos dispositivos, sin centrarnos en uno u otro modelo.

A continuación, vamos a realizar un análisis de los interfaces y mandos de juego más populares entre los usuarios, indicando las diferencias entre unos y otros, y sirviendo de guía a la hora de decidirnos por cualquiera de estos periféricos.

Antes de adquirir un interface para *joystick,* debemos reflexionar sobre el objetivo concreto con el que realizamos la compra. Son varios los factores que hemos de considerar:

— Si tan solo necesitamos un interface sencillo y barato, nuestra elección ha de ir encaminada hacia uno del tipo *Kempston,* puesto que se trata del más frecuentemente empleado en los juegos comerciales.

— Si lo que buscamos es la posibilidad de participación simultánea de dos jugadores, necesitaremos un interface que disponga de salida doble para *joystick.*

— Quizá deseemos un interface compatible con todos los juegos (incorporen o no la opción *joystick);* nuestra elección irá encaminada entonces hacia un interface programable.

— Si nuestro objetivo es disponer de la mayoría de los sistemas de interface empleados comúnmente, existen modelos especiales que vienen preparados para varias opciones *(Sinclair, Kempston, Protek,* etc...).

El precio de los interaces varía, lógicamente, según los modelos y sus prestaciones, siendo habitualmente los menos baratos aquellos que incorporan varias opciones o son programables.

Los interfaces programables son sin duda una buena solución para los poco aficionados al manejo del teclado.

Existe una gran cantidad de interfaces para joystick. *Nosotros deberemos escoger el más adecudo a nuestras necesidades.*

PROGRAMACION DE INTERFACES

Según el sistema de programación de los interfaces, éstos pueden ser divididos aún en dos categorías más: programables por *software* y programables por *hardware*.

Dentro del primer tipo, se puede hacer una nueva subdivisión, según el *software* empleado para la programación del interface se encuentre en una memoria en el interior del mismo, o tenga que ser cargada del exterior via casete.

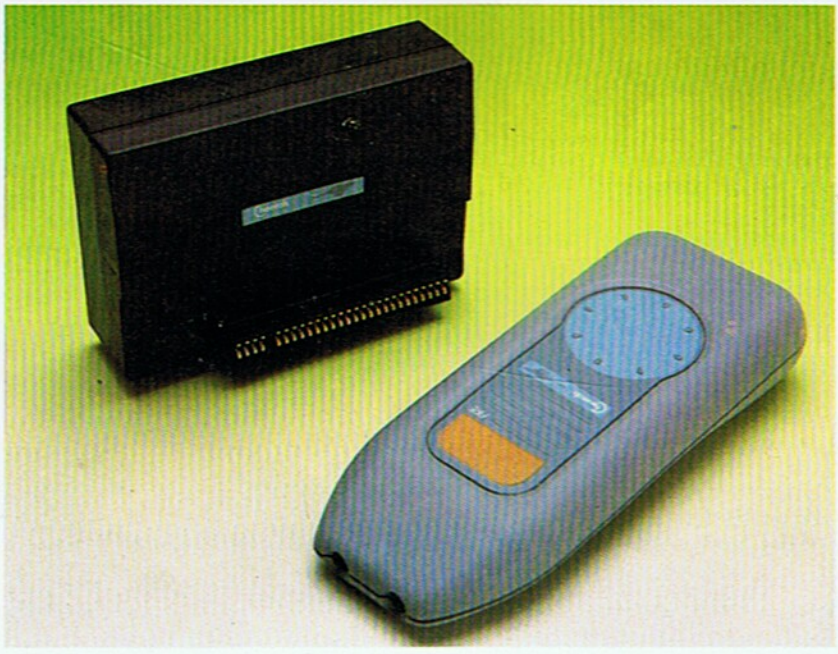
En cualquier caso, una vez cargado el *software* de apoyo pertinente, se puede proceder a la programación del interface; el sistema empleado a tal fin difiere un tanto según los modelos. Algunos de ellos, solicitan mediante una toma de datos directa desde el teclado la indicación de la tecla seleccionada para cada uno de los movimientos. Otros se basan en la investigación de la tecla pulsada, en el momento en que el usuario del interface realiza el movimiento pertinente de la palanca, o pulsa el botón de disparo.

La programación del interface será más o menos laboriosa según los modelos, ya que en unos las diagonales quedan programadas tan solo con definir las direcciones principales, y en otros será necesario ir paso a paso indicando todos los posibles movimientos y si éstos son realizados con o sin disparo.

El sistema de programación *hardware,* como su propio nombre indica, se basa en una acción física sobre el interface. Normalmente, la programación consiste en la presión de determinados botones o la aplicación de clavijas, en un tablero que el interface incorpora simulando el teclado. De esta manera, indicamos cuáles son las teclas que deseamos utilizar para cada juego.

En cuanto a los interfaces no programables, pueden ser de dos tipos: los que sólo admiten una opción (ya sea *Sinclair, Kempston* o cualquier otra) y los que vienen preparados para más de un sistema. Por lo general, estos últimos son capaces de controlar cualquier juego que podamos adquirir.

Vamos ahora a hacer una pequeña visita al mercado de estos aparatos, reseñando las características de los más populares dentro de cada uno de los tipos indicados.



INVESTRONICA SOUND & JOYSTICK INTERFACE

Investrónica comercializa varios tipos de interfaces programables. El de última aparición, nos facilita la posibilidad de escoger mediante un menú

Seguramente, el interface para joystick *más sofisticado, sea el RAT de la firma CHEETAH, accionado por control remoto.*

que aparece en la pantalla entre ocho sistemas diferentes, además de poder programar el interface con las teclas que deseemos. Cuando hayamos seleccionado las direcciones principales, las diagonales quedan programadas de forma automática.

Cuando se ha finalizado la programación del interface, debemos tener cuidado de no desplazar la palanca de juego, ya que se irán marcando sobre la pantalla los valores de las teclas que hayamos escogido. Para solventar este inconveniente, basta con introducir, una vez que estemos en BASIC, la instrucción **OUT 251,4.** Al hacerlo, la palanca queda fuera de uso, sin perder la programación efectuada. Para volver al estado normal, basta con incluir **OUT 251,6.**

Desde BASIC es posible acceder al menú mediante **OUT 251,0.** Hemos de tener en cuenta que si se realiza esta operación, se borrará toda la programación que hubiéramos realizado antes de acceder al menú.

Un dato más sobre el aparato es que lleva incorporado un amplificador de sonido, con un mando que permite regular el volumen.

La firma SINCLAIR ha lanzado al mercado su propio interface para joystick, *el ZX INTERFACE 2. Además de incorporar una entrada* (slot) *para cartuchos de ROM, este interface admite el control de 2 palancas de juego.*

En el mercado del joystick *encontraremos modelos para todos los gustos: desde el popular KEMPSTON 3000, hasta el sofisticado RAT de CHEETAH, pasando por la familia QUICKSHOT de Spectravideo.*

DK'TRONICS

La programación de este interface, se puede realizar de dos maneras distintas. En primer lugar, cargando en memoria un programa que el fabricante suministra junto con el interface, ya sea desde el casete o bien mediante el teclado. Una vez que se encuentra en memoria, el programa se ejecuta y nos va interrogando sobre cuales son las teclas que deseamos escoger para cada dirección, y cual es la seleccionada como botón de disparo. Finalmente, presenta un mensaje sobre si estamos o no de acuerdo con la programación realizada. Si contestamos afirmativamente, el programa se borra de la memoria y el interface queda programado. A continuación podemos ya cargar el juego a utilizar.

El otro sistema es de programación manual y puede realizarse antes o después de la carga del juego. La ventaja de hacerlo así, consiste en que en la mayoría de los juegos la programación puede ser hecha incluso cuando el programa está funcionando. Para ello seguiremos los siguientes pasos:

1) Colocamos el interruptor de la parte superior del interface en la posición 2.

2) Ahora, si lo deseamos, podemos cargar el juego. Si se autoejecuta, lo ignoraremos hasta finalizar la programación.

El interface AGF PROTOCOL 4, programable por **hardware**, *es de una gran sencillez de manejo.*

3) Movemos el *joystick* en todas las direcciones, incluyendo las diagonales y, a continuación, repetimos la operación, pero con el botón de disparo pulsado. Esto sirve para limpiar las posiciones de memoria que el interface va a utilizar.
4) Desplazamos la palanca hacia arriba. Pulsamos la tecla seleccionada para esta dirección, sin soltarla. Liberamos el *joystick* y, acto seguido, la tecla. Este procedimiento lo repetiremos para las cuatro direcciones y el botón de disparo.
5) Colocamos el interruptor en la posición 1. Ahora el interface está ya programado y, si todavía no hemos cargado el juego, podemos ver impresas en la pantalla, según desplacemos la palanca, las teclas seleccionadas. Colocando de nuevo el interruptor en la posición 2, estaríamos en condiciones de comenzar, otra vez, la programación.

El SOUND & JOYSTICK INTERFACE de Investrónica, proporciona además de un cómodo sistema de programación, un amplificador de sonido con volumen regulable.

AGF PROTOCOL 4

Quizás éste sea uno de los modelos más originales que se puedan encontrar en el mercado, a la vez que sencillo de programar. Se trata de un interface programable por *hardware;* dispone, además, de un botón de RESET que nos permite, cuando lo deseemos, apagar y encender el ordenador.
Un conmutador situado junto al botón de RESET, nos permite convertir el interface en compatible *Kempston,* lo que resulta especialmente útil en aquellos juegos que disponen de una rutina para detectar este sistema. El cambio para pasar de *Kempston* a programable puede realizarse en cualquier momento, mediante una nueva acción sobre el interruptor indicado.
La programación del interface se lleva a cabo, por medio de unas tarjetas de plástico negro, que llevan perforadas 5 filas de 8 agujeros cada una y que representan las 40 teclas del teclado. Cuando seleccionamos una tecla, debemos introducir en dos de los orificios unos pequeños conos de plástico semiblando. El lugar donde se colocan, queda determinado mediante los códigos señalados en la tabla grabada sobre el interface.
Las tarjetas para los principales sistemas vienen prefabricadas; pero podemos construir una tarjeta propia con las teclas que queramos seleccionar.

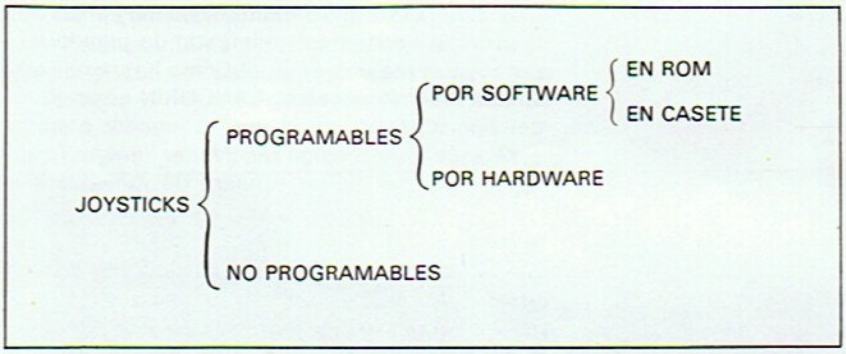
SINCLAIR ZX INTERFACE 2

Este modelo, simula para los movimientos del joystick las pulsaciones de las teclas del 1 al 5,

	NO PROG.	PROG. SOFTWARE	PROG. HARDWARE	KEMPSTON	SINCLAIR	PROTEK	2 SALIDAS	RESET	CARTUCHO
STONECHIP		Mvto. palanca		X	X	X			
AGF PROTOCOL 4			X	X	X	X		X	
INVESTRONICA		Elecc. teclas		X	X	X			
DK'TRONICS		Casete							
RAT CHEETAH	X			X					
INTERSTATE 31	X			X				X	
SINCLAIR	X				X		X		X
RAM TURBO	X			X	X	X	X		X
RAM MK2	X			X					

Cuadro de características de los interfaces para joysticks, *de mayor aceptación.*

y del 6 al 0. Dispone, por tanto, de dos salidas para *joystick,* que para transmitir la información utilizan los ports 61438 y 63486.
Además, lleva incorporada una ranura para cartuchos de ROM, que proporcionan una carga del programa prácticamente instantánea. Al igual que cuando conectamos el interface al ordenador, cada vez que introduzcamos un cartucho tendremos especial cuidado en mantener desconectado de la corriente nuestro Spectrum.
Otro modelo similar al anterior, que también admite cartuchos de ROM, es el RAM TURBO. Incorpora igualmente salidas para dos *joysticks,* aunque a diferencia del *Sinclair* ofrece también la compatibilidad con el sistema *Kempston* y el *Protek.*



Los joysticks *se pueden dividir en varios tipos, según sus características fundamentales, independientemente del sistema de interface que adopten.*

INTERSTATE 31 Y RAM MK2

Para terminar, dos modelos compatibles con *Kempston:* INTERSTATE 31 y RAM MK2. El primero, mediante un conmutador colocado en la parte superior, admite la opción de disparo continuo. Puede ser una buena solución para aquellos juegos en los que nuestro afán por masacraz marcianitos o naves enemigas, no permita ni un momento de respiro y, prácticamente, no podamos despegar el dedo del botón de disparo. El segundo, es un típico interface compatible con *Kempston* que, como solución económica, siempre debe ser considerado.
Del mismo estilo que el RAM MK2, existen multitud de modelos de absoluta compatibilidad con el sistema *Kempston.* Muchos de ellos son versiones «caseras», pero perfectamente válidas, realizadas por algunas tiendas especializadas en artículos microinformáticos.

El interface programable DK'TRONICS, facilita su software *de apoyo en una casete.*

CANGURIN

EL programa de esta semana es una adaptación del conocido juego de mesa "el salto" *(LEAP-FROG* en su origen inglés). Quizás por el momento no nos acordemos de él, pero seguro que al ver funcionar CANGURIN, nos vendrán a la cabeza las innumerables veces que hemos jugado al *LEAP-FROG*.

Este juego tiene una caracteristica muy importante: admite un número indefinido de jugadores... pero eso sí ¡de uno en uno! Como habremos adivinado los más sagaces, CANGURIN es un juego del tipo solitario, en el que el jugador ganador será aquel que consiga resolver el "enigma marsupial" en un menor número de movimientos.

EL ENIGMA MARSUPIAL

El algún perdido paraje australiano, habitan dos tribus de canguros muy especiales. Su peculiaridad reside en el pigmento de su piel: unos son ro-

Ningún canguro podrá saltar nunca sobre una posición ocupada por otro.

sas y los otros azules. Teniendo conocimiento del extraño fenómeno, un grupo de investigadores capturó a los marsupiales para su estudio, pero al retornarlos a su lugar de origen, cometieron un pequeño error. Situaron a los canguros azules en el territorio de los rosas, y a éstos en el de los azules.

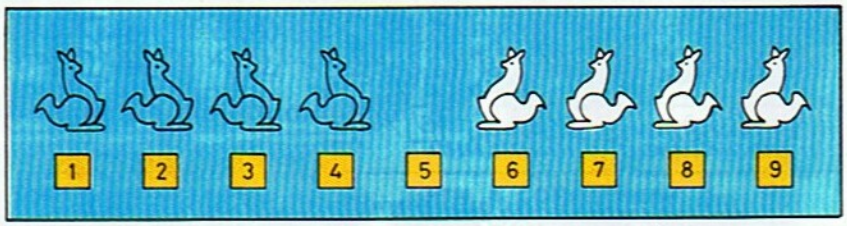
Ambas tribus decidieron, a un mismo tiempo, emprender el éxodo hacia sus hogares. Cuando llevaban recorrido aproximadamente la mitad del camino, y discurrían por una estrechísima garganta rocosa, se toparon con un problema; para ser exactos, con cuatro problemas. Se encontraron con que los canguros de la otra tribu habían llegado por el mismo camino y taponaban el paso. Como eran unos animales muy tozudos, ninguno quería ceder el paso a sus congéneres, de manera que los jefes de tribu tomaron una decisión: había de buscar una solución, para que todos pudieran pasar sin necesidad de que ninguno retrocediera hasta el comienzo de la garganta.

Después de un tiempo, llegaron a un acuerdo sobre cómo debían hacerlo, pero ¿cómo realizaron sus movimientos para conseguir el mínimo esfuerzo?, ¿cuántos movimientos precisaron para poder proseguir su camino? He aquí el enigma marsupial.

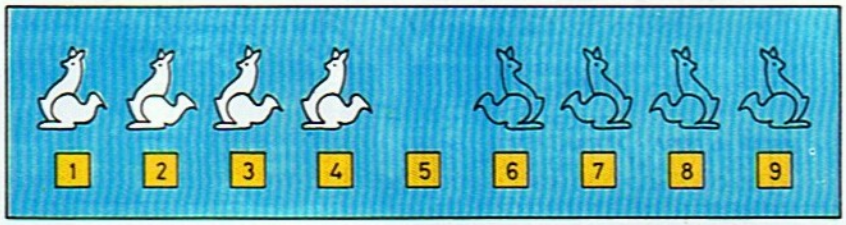
Una vez puestos al corriente del significado del enigma marsupial, va siendo hora de que cada uno de los jugadores intentemos resolverlo a nuestro manera. A los efectos de poder declarar un vencedor final, que podrá ser condecorado con la "Gran Medalla de la Orden de los Marsupiales", reproducida en estas páginas, el programa incorpora un contador de jugadas.

Para jugar a CANGURIN hay que atenerse a muy pocas reglas, y sólo es condición indispensable una cierta dosis de astucia. Siempre que queramos hacer saltar un canguro a alguna posición, tendremos que comprobar si el espacio de destino está libre, es decir, si no está ocupado por otro marsupial de la misma o diferente raza.

Por otra parte, la potencia de salto de los canguros no les permite ir más allá de dos posiciones consecutivas. Como todos sabemos, una imagen vale más que mil palabras, por tanto, fijémonos en las figuras adjuntas para retener las reglas del juego.



El juego comienza con los canguros en la posición de su encuentro en la garganta rocosa.



Ganará la partida el jugador que consiga situar los marsupiales en su posición final, con un menor número de jugadas.

BITS

Mediante el POKE 23609,100 de la línea 80, conseguimos "sonorizar" el teclado del Spectrum, de manera que hacemos más agradable la toma de datos del programa.

*

Recordemos que las letras que aparecen subrayadas en las líneas 170 y 190 del listado, corresponden a los gráficos definidos de las mismas teclas.

*

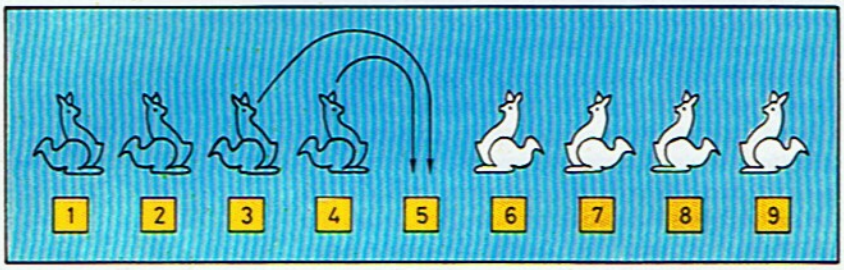
El programa puede ser grabado con autoejecución, mediante **SAVE "CANGURIN" LINE 10.**

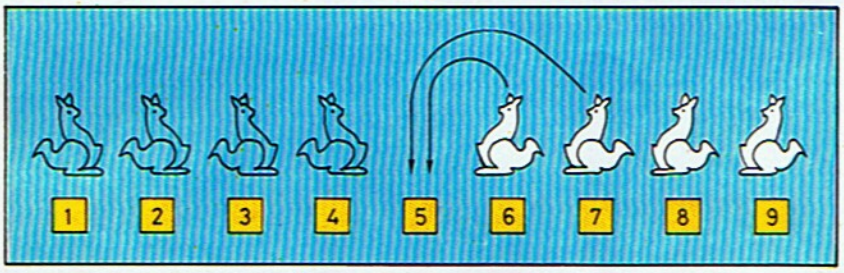
En resumen, existen solamente dos normas a cumplir:

1) Ningún canguro podrá saltar nunca sobre una posición ocupada ya por otro congénere.

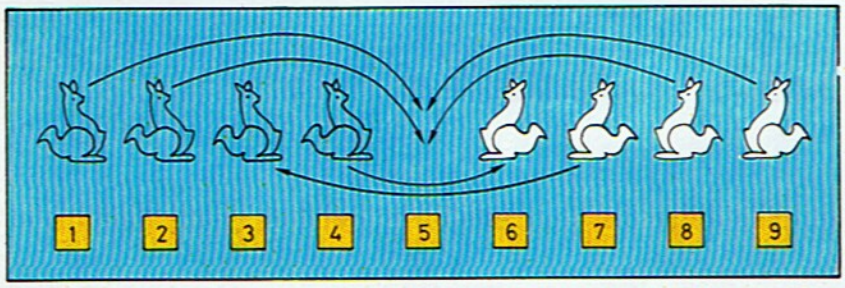
2) Los marsupiales podrán realizar sólo dos tipos de salto; o a la posición contigua o a la siguiente a ésta, pasando por encima del compañero situado a su lado, siempre y cuando las posiciones de destino estén vacías.

Para llevar a cabo el salto, es necesario indicar primeramente en que lugar se encuentra el canguro a desplazar, y a continuación cual es su posición de destino. Si el movimiento es correcto, el ordenador alterará la disposición "canguril" e incrementará en uno el contador de jugadas, de no ser así, volverá a pedir las coordenadas de desplazamiento.





Dos son únicamente los tipos de movimiento admitidos por el programa.



Las reglas del juego descartan muchas de las posibles soluciones al enigma, sobre todo debido a la escasa potencia de salto de los saltarines mamíferos.

```
 10 REM     ****************
 20 REM     ****************
 30 REM     ** J.M.MAYORAL **
 40 REM     ****************
 50 REM     ****************
 60 BORDER 6: PAPER 6: CLS : CLEAR 32399
 70 GO SUB 840
 80 POKE 23609,100: POKE 23658,0
 90 LET D$=" DATOS INCORRECTOS "
100 INK 9
110 LET A=1: LET B=7: LET SW=0
120 LET C=11: LET D=17
130 LET CONT=1: LET JUG=0
140 FOR N=3 TO 8
150 PRINT  PAPER 1;AT N,2;"
160 NEXT N
170 LET P$="A A A A   B B B B"
180 LET Q$=P$
190 LET O$="Q Q Q Q   P P P P"
200 LET W$=O$
210 FOR N=3 TO 19 STEP 2
220 PRINT  PAPER 6;AT 7,N;CONT
230 LET CONT=CONT+1
240 NEXT N
250 PRINT AT 11,18;"JUGADAS= "
260 PRINT  PAPER 1;AT 5,3;Q$;AT 4,3;W$
270 IF Q$(A TO B)=P$(C TO D) AND Q$(C TO D)=P$(A TO
B) THEN  GO TO 410
280 INPUT "Que canguro vas a desplazar?""A$
290 IF LEN A$<>1 OR CODE A$<=48 OR CODE A$>=58 THEN
GO SUB 640: GO TO 280
300 INPUT "A donde va a saltar?""B$
310 IF LEN B$<>1 OR CODE B$<=48 OR CODE B$>=58 THEN
GO SUB 640: GO TO 280
320 LET K$=A$+B$: LET S$=O$+W$
330 LET X=2*(CODE K$(1)-48)-1
340 LET Y=2*(CODE K$(2)-48)-1
350 IF Q$(Y)<>" " OR ABS (Y-X)>4 THEN  GO TO 280
360 LET JUG=JUG+1
370 PRINT  PAPER 2;AT 11,27;JUG
380 LET Q$(Y)=Q$(X): LET W$(Y)=W$(X)
390 LET Q$(X)=" ": LET W$(X)=" "
400 GO TO 260
410 GO SUB 770
420 LET D$=" LO CONSEGUISTE MUCHACHO "
430 LET K=7: LET SW=1
440 GO SUB 650
450 LET D$=" BRAVO ": LET K=2
460 FOR W=16 TO 20
470 FOR Z=10 TO 17
480 GO SUB 660
490 NEXT Z
500 NEXT W
510 FOR N=1 TO 255
520 OUT 254,N
530 NEXT N
540 CLS : PRINT AT 20,0;"DESEAS INTENTARLO OTRA VEZ?
(s/n)"
550 IF INKEY$="s" THEN  RUN
560 IF INKEY$="n" THEN  GO TO 580
570 GO TO 550
580 LET D$="TU TE LO PIERDES DESAGRADECIDO"
590 LET W=14: LET Z=0: LET K=6
600 GO SUB 660
610 PAUSE 100: BEEP .2,30: BEEP .2,10
620 STOP
630 REM RUTINA DE FALLOS
640 LET K=2
650 LET W=14: LET Z=5
660 FOR N=1 TO LEN D$
670 PRINT  PAPER K;AT W,Z;D$(N): BEEP .01,31
680 LET Z=Z+1
690 NEXT N
700 IF SW=1 THEN  RETURN
710 PAUSE 100
720 FOR N=1 TO LEN D$
730 PRINT  PAPER 6;AT 14,N+4;" ": BEEP .01,15
740 NEXT N
750 RETURN
760 REM RUTINA SONIDO
770 RESTORE
780 DATA 6,5,197,33,0,3,17,1,0,229,205,181,3,225,17,
16,0,167,237,82,32,240,193,16,233,201
790 FOR P=32400 TO 32425
800 READ B: POKE P,B
810 NEXT P
820 RANDOMIZE USR 32400
830 RETURN
840 REM GRAFICOS USUARIO
850 RESTORE 860
860 DATA BIN 10000100,BIN 10100100,BIN 10001011,BIN
01001010,BIN 00101000,BIN 01010000,BIN 10100110,BIN 1
1111111
870 DATA BIN 00100000,BIN 01100000,BIN 01010000,BIN
01001111,BIN 01011110,BIN 01100000,BIN 10100000,BIN 1
0010000
880 FOR N=0 TO 7
890 READ F: POKE USR "A"+N,F
900 NEXT N
910 FOR N=0 TO 7
920 READ F: POKE USR "O"+N,F
930 NEXT N
940 FOR N=0 TO 7
950 READ G: POKE USR "B"+N,G
960 NEXT N
970 DATA BIN 00111111,BIN 00111011,BIN 11010111,BIN
01111110,BIN 00011100,BIN 00001110,BIN 01110110,BIN 1
1111111
980 DATA BIN 00000100,BIN 00000110,BIN 00001010,BIN
11111110,BIN 01111110,BIN 00000110,BIN 00000111,BIN 0
0001111
990 FOR N=0 TO 7
1000 READ G: POKE USR "P"+N,G
1010 NEXT N
1020 DATA 32,32,74,46,77,46,77,65,89,79,82,65,76
1030 FOR C=7 TO 19
1040 READ F
1050 PRINT  INK 9;AT 0,C;CHR$ F
1060 NEXT C
1070 RETURN
```

PRINCIPIOS DE PROGRAMACION EN BASIC

PARA comenzar, debemos decir que el BASIC es un lenguaje de alto nivel, es decir, no excesivamente difícil de manejar y comprender por nosotros. Por medio de un conjunto no muy grande de instrucciones, podemos realizar programas que resuelven problemas muy variados e incluso complicados.

Hasta ahora, hemos trabajado en el modo directo o comando. Con este sistema hemos visto que las instrucciones que introducíamos a través de la línea 24 de la pantalla del ordenador, se ejecutaban de forma inmediata al pulsar la tecla **ENTER**.

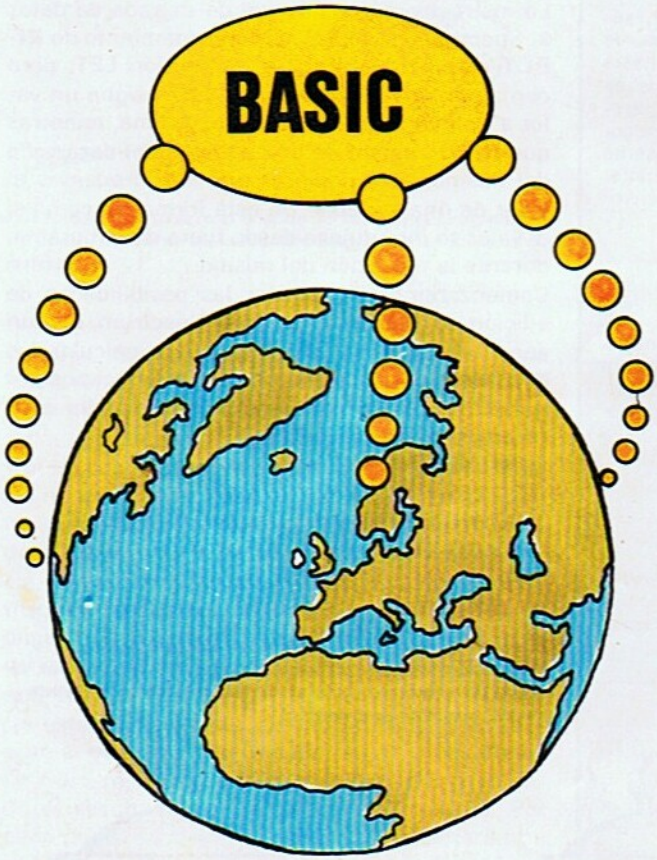
Para consultas rápidas sobre el valor o de una variable, o para averiguar el resultado de una operación matemática, el modo directo cumple perfectamente su cometido; pero cuando se trata de resolver problemas más complicados, que requieren de una entrada de datos, la realización de ciertos cálculos y una salida impresa, la utilización de este modo es imposible.

El modo PROGRAMA existe, precisamente, para facilitar la construcción de estructuras de programación más complejas, que permitan almacenar gran número de instrucciones, guardando un determinado orden de ejecución. Un programa BASIC no es más que una colección de instrucciones, numeradas para indicar al Spectrum el orden exacto en que deben ser ejecutadas.

Cuando anteponemos un número a lo que sería una instrucción en modo directo, el ordenador entiende que no la debe ejecutar de forma inmediata, sino almacenarla dentro del programa y colocarla en la posición que le corresponde.

Normalmente, las instrucciones de programa se numeran de 10 en 10. Esto se hace para permitir intercalar nuevas instrucciones entre las ya escritas. Cada vez que el Spectrum acepta una nueva línea para el programa, lo primero que se pregunta es si ese número de instrucción ya existe. De ser así, reemplaza la antigua línea por la nueva que le ha llegado con el mismo número. Si la instrucción no existía antes, la incluye dentro del programa, entre las líneas a que corresponde de acuerdo con su número.

El Spectrum nos permite, a través de su línea de edición (la última), la introducción, modificación y borrado de líneas de programa; contando para ello con la ayuda de la función **EDIT**, (**SHIFT** y **1** en el modelo antiguo), y las teclas de desplazamiento del cursor.



El BASIC es el lenguaje de programación, más ampliamente difundido en el mundo microinformático.

COMENZANDO A PROGRAMAR

Dado que el repertorio de palabras BASIC que manejamos es bastante limitado, vamos a ampliarlo algo en este capítulo. Un programa cons-

Los lenguajes de programación más utilizados en el Spectrum son el BASIC, el ensamblador y el código máquina. De entre ellos, el BASIC es el de más alto nivel, es decir, el de mayor facilidad de comprensión por los humanos.

El *syntax marker*, señala en la línea de edición el punto hasta el cual la sintaxis de la instrucción a introducir se considera correcta. Se representa mediante un carácter de interrogación en intermitencia.

La zona de la pantalla que se emplea para la introducción y modificación de las instrucciones, se denomina LINEA DE EDICION.

El comando fundamental para la modificación de programas es **EDIT**.

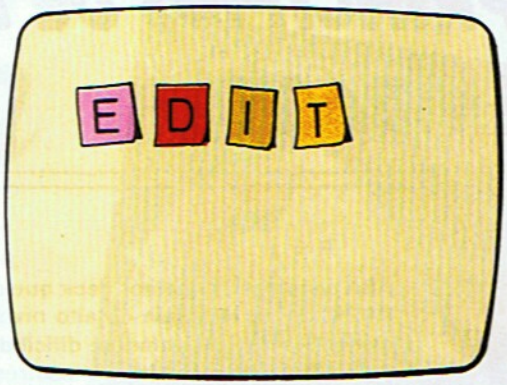
ta de tres partes fundamentales: entrada, cálculo y salida. Conocemos ya algunas palabras BASIC que nos permiten efectuar la salida de resultados (**CLS, PRINT**), y operadores aritméticos con los que realizar los cálculos (+, –, *, /). Nos falta, por tanto, conocer alguna instrucción para introducir datos en el ordenador.

La instrucción fundamental de entrada de datos al Spectrum es **INPUT**. El funcionamiento de **INPUT** es parecido al de la instrucción **LET**, pero con algunas particularidades. **LET** asigna un valor a una variable dentro del programa, mientras que **INPUT** establece una pausa en el desarrollo del mismo, para pedirnos que introduzcamos el valor de una variable. De esta forma, es como si el valor se introdujese desde fuera del programa, durante la ejecución del mismo.

Comenzaremos a practicar las posibilidades de edición de instrucciones en el Spectrum, con un sencillo programa. En él, vamos a calcular los cuadrados de los números que introducimos por un **INPUT**; para mostrarlos en la pantalla utilizaremos la instrucción **PRINT**:

```
10 INPUT N
20 LET C=N*N
30 PRINT "El cuadrado de";N;" es:";C
40 GO TO 10
```

El modo directo, da vía libre a la ejecución de las instrucciones según entran en el ordenador. En cambio, el modo programa pospone su ejecución hasta el momento en que nosotros lo deseemos.



El comando EDIT *se utiliza para editar cualquier línea de programa en el área de modificación (zona inferior de la pantalla).*

Debemos recordar pulsar **ENTER** después de cada nueva línea de instrucción. Veremos que, según vamos introduciendo las líneas, éstas en vez de ejecutarse de forma inmediata, pasan a formar parte ordenada del programa y son transferidas a la parte superior de la pantalla. Hagamos una pequeña revisión de los conocimientos empleados en el programa.

Punto y coma (;) es un parámetro de la palabra clave **PRINT**, que actúa como separador entre variables y literales, indicando a un mismo tiempo que el siguiente dato impreso en la pantalla, debe situarse inmediatamente después del que ha precedido al punto y coma. **GO TO** es una instrucción que transfiere el control del programa a la línea que indica.

Una vez que hemos almacenado el programa, tenemos que ejecutarlo, o sea, hacerlo funcionar. Esto se logra empleando la palabra clave **RUN** en modo directo. **RUN** borra los contenidos de las variables que pudieran existir con anterioridad al inicio del programa, limpia la pantalla y comienza a ejecutar el programa a partir de su primera línea.

De no existir en el programa ninguna sentencia de transferencia de control (**GO TO**), el programa pasaría por las líneas 10, 20, y por último 30, para detenerse a continuación por haber llegado al final del programa. Esto no sucede, porque el **GO TO** de la línea 40, indica al ordenador que el programa debe continuar a partir de la instrucción 10. Todos los programas están formados por una secuencia de instrucciones numeradas, de forma que al teclear **RUN** el ordenador ejecuta una a una las líneas que lo componen, por riguroso orden, hasta llegar a la última, donde se detiene.

La instrucción **GO TO** es una declaración imperativa. Esto quiere decir, que no está condicionada a ninguna otra circunstancia y que, cada vez que el programa pase por ese número de línea, debe continuar la ejecución a partir del nuevo número de instrucción que se le indica.

En nuestro ejemplo, al teclear **RUN** (palabra clave de la tecla **R**), el programa comienza a ejecutar la primera línea (10), demandándonos que introduzcamos un número. En la última línea de la pantalla, aparece el cursor en el modo **L**, para indicarnos que espera una secuencia de caracteres; debemos entonces teclear el número del que deseamos calcular el cuadrado, seguido de **ENTER**.

Hecho esto, el Spectrum pasa a interpretar el siguiente número de instrucción (20). Asigna a la variable numérica C, por medio de la instrucción **LET**, el resultado del producto del número introducido por nosotros en el **INPUT N**, por sí mismo; es decir, asigna a C, el valor N al cuadrado. Una vez realizado el cálculo, continua con la interpretación de la siguiente instrucción (30), que le indica como mostrar los resultados obtenidos en la pantalla. Como el programa no ha escrito aún ningún texto, la instrucción **PRINT** comienza a ejecutarse, a partir de la primera columna de la primera línea de la pantalla.

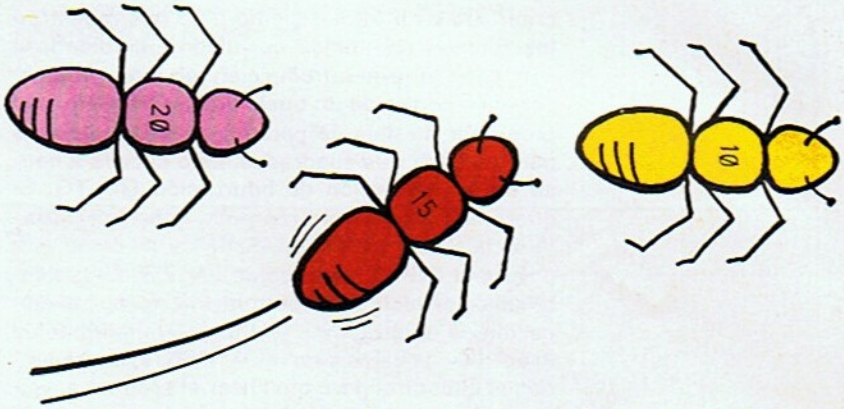
Lo primero que se escribe es el literal (la secuencia de caracteres entrecomillados) "El cuadrado de". Después imprime el valor de la variable N (inmediatamente a la derecha del literal por existir un ;), seguidamente el literal " es:", e inmediatamente el resultado almacenado en la variable C.

El **INPUT** *es como un buzón del programa, en el que el usuario puede depositar los datos con destino a éste. Dispone de dos entradas diferentes: una para variables numéricas y otra para variables de cadena.*

Observemos que, hasta aquí, toda la instrucción **PRINT** ha estado condicionada por el punto y coma (;), por lo que cada nuevo módulo de la misma se ha colocado a la derecha del anterior. Como al final de la instrucción (después de C), no aparece este signo de puntuación, el programa interpreta que las próximas instrucciones **PRINT** deben ejecutarse en la siguiente línea de la pantalla, por lo que se sitúa en ella cediendo el control de programa a la siguiente instrucción (40). De no existir esta línea de instrucción, el programa detendría aquí su ejecución, al haber rebasado el número de instrucción más alto en secuencia. Sin embargo, como hemos incluido una instrucción de bifurcación incondicional (**GO TO**), al llegar el programa a la línea 40, lo que hace es volver a comenzar con la interpretación de la línea 10.

De esta forma hemos creado un «bucle» de programa, que se repite una y otra vez, pidiéndonos un número, calculando su cuadrado e imprimiendo el resultado del cálculo en la pantalla. Por cada pasada por la instrucción 30 se genera una nueva línea de resultados.

Después de llenar con los resultados la primera pantalla, observamos que el ordenador efectua un *scroll* de la misma. Esta función automática, suprime la línea superior de la pantalla y desplaza las demás una línea hacia arriba, para dejar sitio en la última a la nueva línea que se va a es-



Las líneas de instrucción ocupan su sitio en el programa, no por su orden de introducción, sino según su número de instrucción.

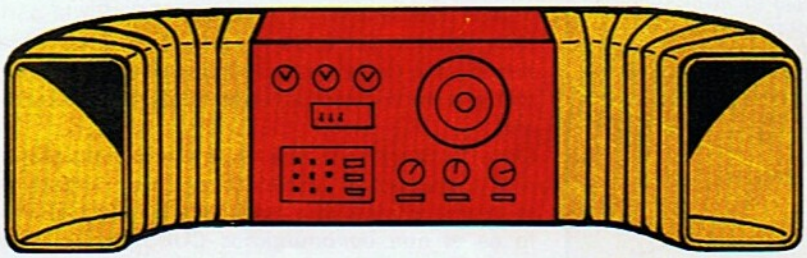
¡!

Cuando una instrucción es escrita en la línea de edición sin ser precedida por un número, se ejecuta inmediatamente al pulsar **ENTER**. A este tipo de instrucciones se les denomina de MODO DIRECTO o MODO COMANDO.

*

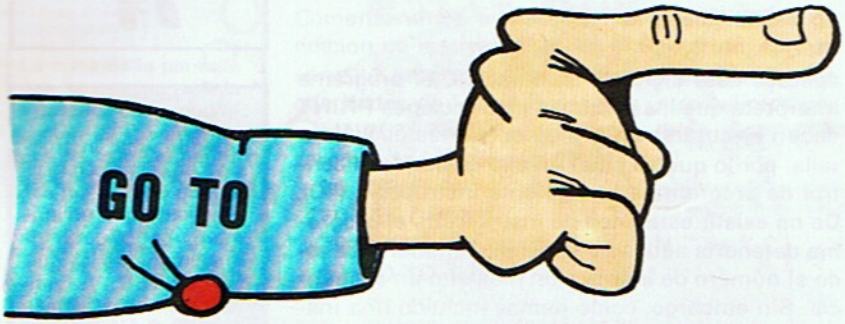
El MODO PROGRAMA facilita la construcción de estructuras de programación más complejas, que permiten guardar un gran número de instrucciones siguiendo un determinado orden de ejecución.

El proceso seguido por un programa, se desarrolla en tres fases: entrada de datos, cálculo en base a los mismos y salida de los datos calculados.



ENTRADA CALCULO SALIDA

cribir. De esta forma, siempre se nos muestran los últimos resultados calculados, perdiéndose con cada nueva introducción el más antiguo. Estamos dentro de un bucle que no tiene fin, y el programa no deja de pedirnos nuevos números para calcular sus cuadrados. Esto sucede a causa de la instrucción de bifurcación **GO TO**. Si queremos detener el programa, debemos pulsar la tecla **STOP** (**SYMBOL SHIFT** y **A**) como respuesta al **INPUT**, seguida de **ENTER**. Esta combinación de teclas, nos permite interrumpir la ejecución de un programa BASIC en el momento de la entrada de datos a través de **INPUT**, devolviéndonos el control para modificar el programa, volver a ejecutarlo, o suprimirlo de la memoria para comenzar a introducir otro nuevo.



La sentencia **GO TO** *obliga a continuar la ejecución del programa, a partir de determinado punto del mismo, que se señala en forma de número de línea a continuación de la sentencia.*

¡!

La instrucción fundamental de entrada de datos al Spectrum es **INPUT**. Esta sentencia establece una pausa en el desarrollo del programa, para que introduzcamos desde el teclado el valor de una variable determinada por el mismo. **RUN** borra las variables que pudieran existir con anterioridad al inicio del programa, limpia la pantalla y comienza a ejecutarlo a partir de su primera línea.

MODIFICANDO NUESTRO PRIMER PROGRAMA BASIC

Vamos ahora a detener el programa para incluir en él algunas modificaciones. Lo primero que hacemos, después de abandonar el **INPUT** por el procedimiento explicado en el párrafo anterior, es pulsar nuevamente la tecla **ENTER**. En la pantalla aparece el listado del programa que hemos introducido (la relación de instrucciones ordenadas con su correspondiente número de secuencia).

Observamos ahora, que en la instrucción número 40, aparece el símbolo > entre el número de la misma y la palabra clave **GO TO**. Este símbolo es el que denominamos CURSOR DE PROGRAMA.

Este cursor de programa apunta siempre al último número de instrucción introducido, modificado o listado. En nuestro caso, está presente en la línea 40 porque ha sido la última que hemos introducido.

Podemos probar ahora a desplazar este cursor por medio de **SHIFT** y las teclas **6** y **7** (flechas verticales en el Plus). Vemos que por cada pulsación de las teclas **SHIFT** y **7** (flecha hacia arriba), el cursor se desplaza una línea hacia arriba, hasta llegar al número más bajo (10). Del mismo modo, empleando ahora las teclas SHIFT y **6** (flecha hacia abajo), el efecto es el contrario, hasta llegar nuevamente a la línea 40.

Nos vamos a situar ahora en la línea 10, para introducir una pequeña modificación. Queremos que, además de que el ordenador nos pida un número, nos indique con un texto la palabra "Número:", aclarando qué tipo de dato se intenta recoger en el **INPUT**. Para poder hacer esto, basta con colocar un texto entre comillas entre la palabra clave **INPUT** y la variable N.

Hasta ahora la instrucción tenía el aspecto:

10 INPUT N

Para situarnos en la línea 10, vamos a emplear los cursores de programas y a teclear **SHIFT** y **1** (**EDIT**). Ahora podemos ver que en la última lí-

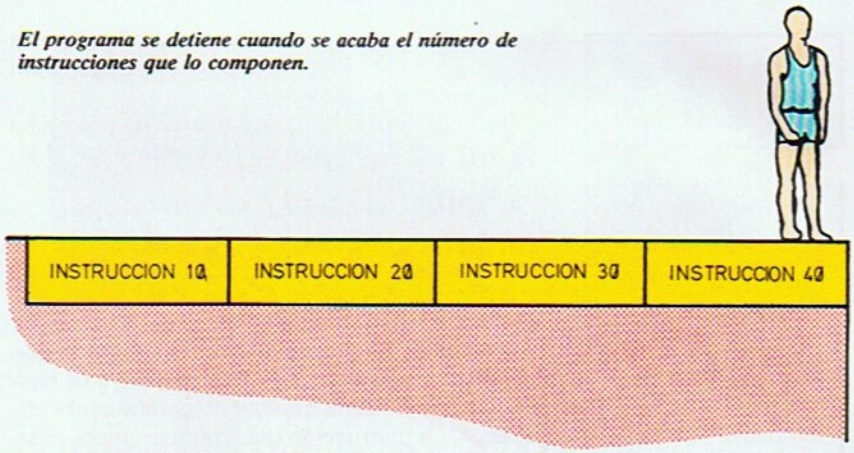
El comando **RUN** *se utiliza para ejecutar las instrucciones, que han sido almacenadas en forma de programa.*

nea de la pantalla se encuentra una copia de la instrucción número 10, lista para ser modificada, con el cursor en el modo **L** inmediatamente situado entre el número de instrucción y la primera palabra clave BASIC.

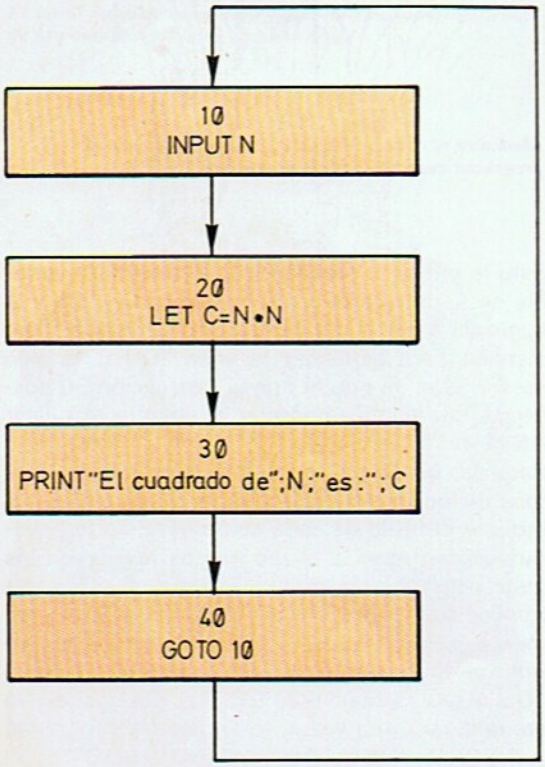
Esta vez, debemos emplear las teclas de desplazamiento horizontal del cursor **SHIFT 5** y **8** (**flechas horizontales**), para colocarnos en el lugar adecuado e introducir la modificación. Pulsamos **SHIFT** y **8** para colocarnos entre **INPUT** y **N**, y tecleamos **"Numero:";**. Según hemos ido introduciendo el literal, la **N** que quedaba a su derecha, se ha ido desplazando para dejar sitio a los caracteres recién llegados. Seguidamente, pulsamos **ENTER** para validar la instrucción (no importa en qué punto se encuentre el cursor al pulsar **ENTER**), y podemos ver como el nuevo contenido de la instrucción 10 ha reemplazado al anterior, desapareciendo la línea de edición.

Podemos ejecutar nuevamente el programa, por medio de la palabra clave **RUN**, para ver los resultados. Efectivamente, el programa funciona de la misma manera que antes, pero pidiéndonos los números a introducir con el texto deseado. Para interrumpir la ejecución debemos emplear el sistema descrito anteriormente, pulsando **ENTER** una vez más si queremos listar el programa. Vamos a continuar ahora con las modificaciones, para conseguir que se imprima una cabecera general para toda la pantalla, en dos columnas denominadas "NUMERO" Y "CUADRADO". Pero para poder hacer ésto, necesitamos saber como escribir columnas de números de forma fácil. Del mismo modo que el punto y coma (;) le indica a **PRINT**, que lo próximo que escriba debe situarse inmediatamente a continuación, la coma (,) tiene el efecto de producir un salto desde la columna 0 a la 16 en la pantalla, es decir, de la primera a la segunda mitad de la misma, ya que la pantalla del Spectrum tiene 32 columnas. De este modo, tenemos solucionado el problema. Así pues, podemos decir que tanto el caracter punto y coma (;), como la coma (,), actúan como separadores de literales y variables, pero mientras que el primero indica que los resultados de **PRINT** se deben escribir seguidos, el segundo los

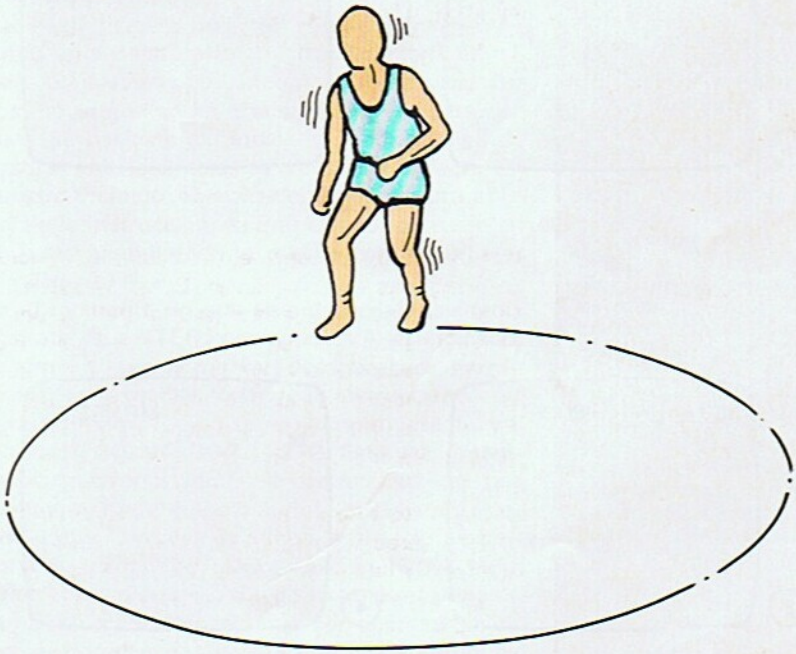
El programa se detiene cuando se acaba el número de instrucciones que lo componen.



El programa «CUADRADOS» sigue el camino señalado por las flechas en el gráfico.



En el programa «CUADRADOS», la instrucción **GO TO 10** *de la línea 40, crea un círculo vicioso en el que se desarrolla el programa. Este tipo de estructuras se denominan BUCLES.*



BITS

Para borrar una línea de instrucción, basta con teclear su número y pulsar **ENTER**.
Para detener un programa durante un **INPUT**, debemos introducir el comando **STOP**.

*

El cursor de programa se desplaza hacia arriba, mediante las teclas **CAPS SHIFT + 7** (tecla especial en el Plus), y hacia abajo, mediante **CAPS SHIFT + 6** (tecla especial en el Plus).

dispone de forma que se escriban en la primera columna o en la mitad de la pantalla, según cual sea el punto más próximo.
Vamos a dirigirnos con los cursores de programa a la instrucción 30, que de momento tiene el siguiente aspecto:

30 PRINT "El cuadrado de";N;" es:";C

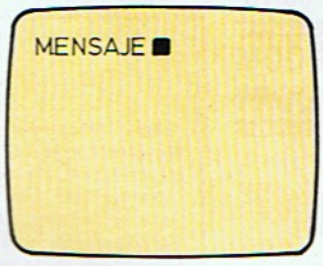
Comencemos por editarla mediante **EDIT**. A continuación, pulsamos **SHIFT** y **8** (cursor derecho) varias veces, hasta que el cursor (en el modo **L**) se sitúe inmediatamente después del ;, que sigue a las comillas finales del literal «El cuadrado de». Hecho esto, pulsamos **DELETE** (**SHIFT** y **0**) hasta que el ; y el literal quedan completamente eliminados. La instrucción debe ofrecer ahora el siguiente aspecto, con el cursor situado inmediatamente antes de la **N**:

30 PRINT N;" ES:";C

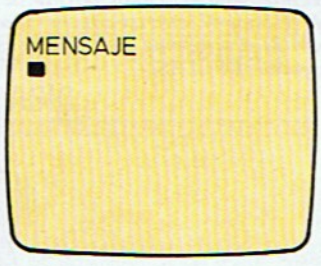
Avanzamos ahora nuevamente el cursor hacia la derecha, para situarnos justamente delante de la **C**. Acto seguido, utilizamos 8 veces la función **DELETE**, para hacer desaparecer el literal que resta y los ;, y tecleamos un caracter ,. Ahora la instrucción debe aparecer como:

30 PRINT N,C

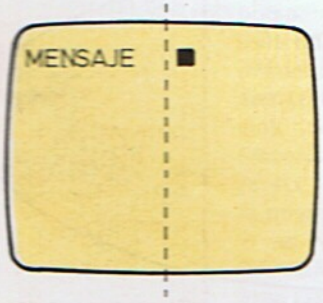
Los separadores punto y coma (;) y coma (,) afectan, a la posición en que se escribirá el próximo literal o resultado (■). I. PRINT "MENSAJE";... *II.* PRINT "MENSAJE"... *III.* PRINT "MENSAJE",... *(cuando la impresión anterior termina en la mitad izquierda de la pantalla) IV.* PRINT "MENSAJE",... *(cuando la impresión anterior termina en la mitad derecha de la pantalla).*



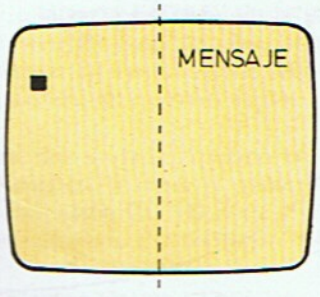
1



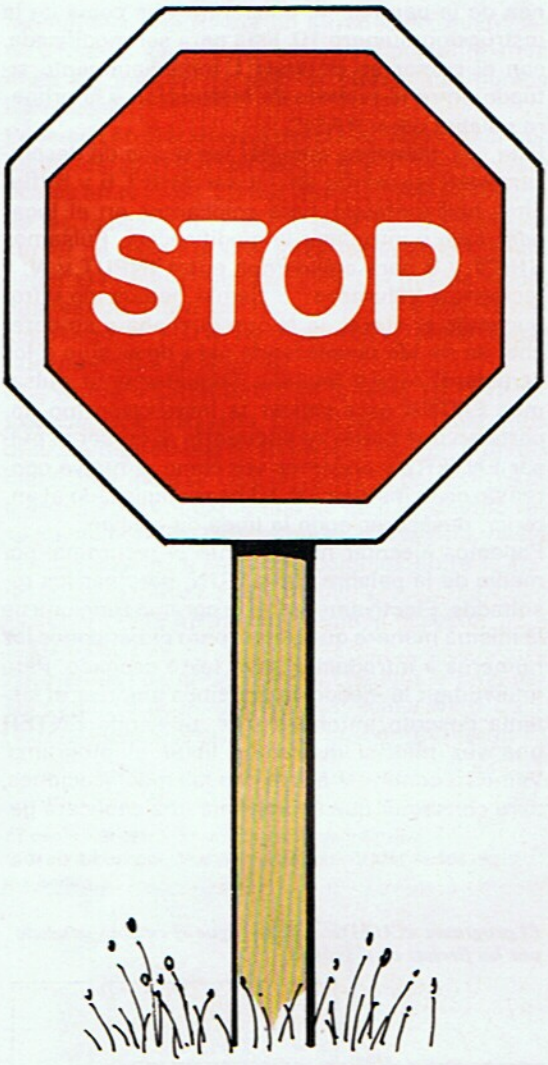
2



3



4



Mediante el comando STOP, *podemos detener el programa cuando se encuentra en un* INPUT.

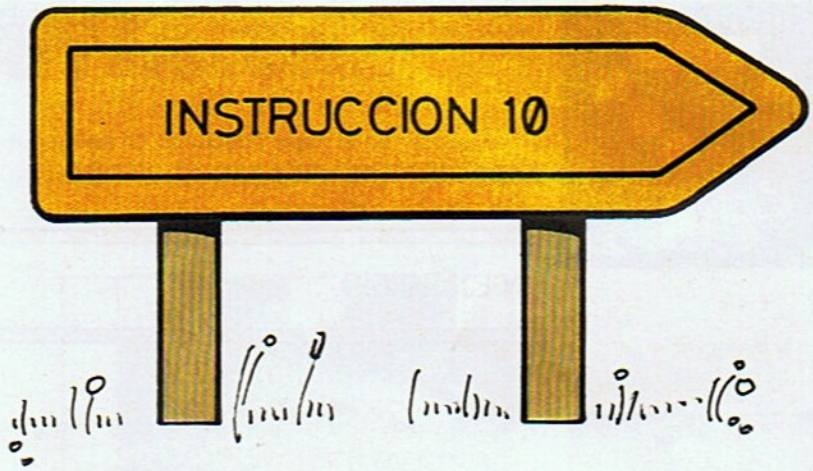
Esto le indica al ordenador que la primera variable se imprimirá al comienzo de cada línea y la segunda a partir del centro de la pantalla. Otro sistema para modificar la línea 30 habría sido reescribirla, ya que la nueva instrucción 30 sustituiría automáticamente a la antigua al pulsar **ENTER**.
Para dar un último toque a nuestra labor, podemos incluir una cabecera en la pantalla que nos indique el título de cada columna. Esto lo podemos hacer, gracias a que hemos numerado las instrucciones de forma que podemos intercalar nuevos números entre ellas. La cabecera la vamos a incluir en la línea 5, justo antes del comienzo de la petición de datos por **INPUT** (línea 10 a la que se efectua el **GO TO**), por lo que sólo se ejecutará una vez al comienzo del programa.

5 PRINT "NUMERO", "CUADRADO"

Después de todas las modificaciones, el programa debe presentar el siguiente aspecto:

```
5 PRINT "NUMERO","CUADRADO"
10 INPUT "Numero:";N
20 LET C=N*N
30 PRINT N,C
40 GO TO 10
```

Ha llegado el momento deseado. Podemos teclear **RUN** para obtener la compensación a nuestros esfuerzos.



El cursor de programa (>) indica la instrucción que aparecerá en la línea de edición al utilizar el comando **EDIT.**

PROBLEMAS DE SINTAXIS

Hasta ahora nos hemos limitado a copiar ejemplos de programas, y por tanto no hemos tenido problemas de ningún tipo; sin embargo, cuando empecemos a hacer nuestros propios programas, la inexperiencia nos hará cometer pequeños errores. El más frecuente es el error de sintaxis, es decir, el que se produce al introducir una instrucción que el ordenador no es capaz de comprender. El Spectrum está dotado de un sistema de detección de errores de este tipo, extremadamente útil para los que estamos aprendiendo. Gracias a él, cualquier instrucción, ya sea en modo directo o programa, no será admitida por el ordenador si contiene algún error de sintaxis. Este sistema se conoce como *syntax checker* (comprobador de sintaxis).

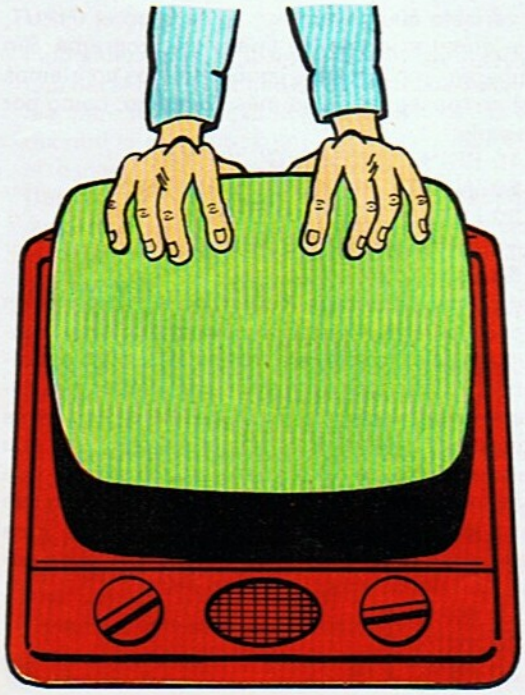
Para estudiar el funcionamiento de este sistema, intentaremos introducir en comando directo una instrucción con un error de este tipo. Por ejemplo, **PRINT "correcto**. Si ahora pulsamos **ENTER**, la línea de edición no desaparecerá, sino que se verá sustituida por la instrucción introducida por nosotros, más un caracter de interrogación en intermitencia, situado entre la instrucción tecleada y el cursor.

¿Qué significa ese caracter? ¿Es un nuevo tipo de cursor que desconocemos? No. Simplemente se trata del sistema por el cual, el *syntax checker* nos indica que se ha producido un error de sintaxis. La posición del indicador, señala el punto hasta el cual la sintaxis se considera correcta. En nuestro ejemplo, el indicador se encuentra al final de la instrucción, ya que el error consiste en no haber completado la misma con el caracter comillas.

En el momento en que se pulse cualquier tecla (distinta de **ENTER**, por supuesto), el indicador de sintaxis (*syntax marker*) desaparecerá, permitiéndonos la modificación de la instrucción.

Para terminar, veamos otro ejemplo. Intentemos introducir **PRINT "2+2= "2+2.** Esta vez, el indicador de error aparecerá entre las comillas que cierran el literal y la constante **2+2.** Esto es debido a que, como ya sabemos, entre los diferentes componentes de una instrucción **PRINT**, debe emplearse un separador (ya sea un punto y coma o una coma). Si retrocedemos con el cursor, la interrogación desaparecerá, y podremos introducir

El scroll consiste en el desplazamiento de la información de la pantalla una línea hacia arriba.



¡!

El separador coma (,), desplaza la posición de la próxima impresión a la primera columna de la siguiente línea, o la columna 16 de esa misma línea, según haya sobrepasado o no esta última. El Spectrum está dotado de un sistema de detección de errores de sintaxis. Gracias a él, cualquier instrucción, ya sea en modo directo o en modo programa, no será admitida por el ordenador si contiene algún error de este tipo. Este sistema se conoce como *syntax checker* (comprobador de sintaxis).

El Spectrum utiliza el syntax marker (?), para indicarnos que no comprende una instrucción.

un separador en el lugar adecuado. Si ahora pulsamos **ENTER** la instrucción será admitida.

UN EJERCICIO

Normalmente, las instrucciones de programa se numeran de 10 en 10. Esto se hace para permitir intercalar nuevas instrucciones entre las ya escritas.

*

Cuando se introduce una nueva línea de instrucción con el mismo número que alguna ya existente, la nueva pasa a sustituir a la antigua en el programa.

*

Cuando anteponemos un número a lo que sería una instrucción en modo directo, el ordenador entiende que no la debe ejecutar de forma inmediata, sino almacenarla dentro del programa y colocarla en la posición que le corresponde.

Por fin ha llegado el momento de ejercitar nuestros conocimientos. Encendamos nuestro ordenador e intentemos diseñar un programa similar al "CUADRADOS" de ejemplo. Nuestro programa ha de calcular la mitad de los números que introduzcamos, es decir, dividir el número introducido por dos. Ahora vamos a ver unas pequeñas pistas que nos ayudarán a resolver el ejercicio: El número a dividir podríamos introducirlo en una variable mediante un **INPUT**. En cuanto al resultado, se puede calcular como la variable entrada dividida por dos, pudiéndose a su vez asignar este valor a una variable. Una vez completada la entrada y el cálculo, sólo resta realizar la impresión en la pantalla del resultado. Ya podemos ponernos manos a la obra; cuando hayamos conseguido un programa que cumpla las premisas anteriormente establecidas, podremos comprobar las diferentes soluciones que aparecen a continuación.

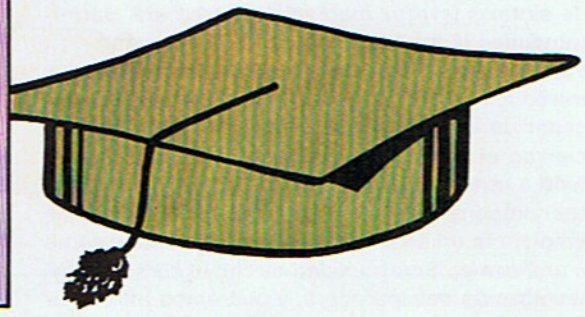
Sin duda, una de las soluciones más simples es:

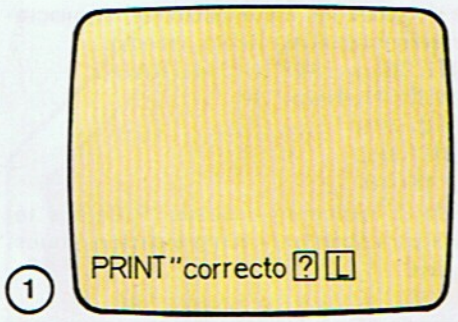
```
10 INPUT A
20 PRINT A/2
```

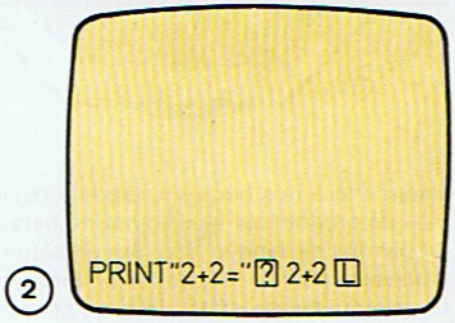
También podíamos habernos esforzado un poquito más, y haber programado algo parecido a:

```
10 INPUT A
20 PRINT "La mitad de ";A;" es ";A/2
```

El sistema comprobador de sintaxis (syntax checker) nos ayudará en la realización de programas, al no admitir líneas que contengan errores de este tipo.







I. Disposición de la pantalla cuando se produce un error de sintaxis, al intentar introducir la instrucción PRINT "correcto.
II. Disposición de la pantalla cuando se produce un error de sintaxis, al intentar introducir la instrucción PRINT "2+2= "2+2.

Por supuesto, no tiene importancia el nombre de la variable que hayamos empleado en el **INPUT**, o la numeración de las líneas del programa. Sin embargo, con nuestros conocimientos podríamos haber conseguido algo más completo, como por ejemplo:

```
10 PRINT "NUMERO","MITAD"
20 INPUT "NUMERO:";NUMERO
30 LET MITAD=NUMERO/2
40 PRINT NUMERO,MITAD
50 GO TO 20
```

Para estar contentos, deberíamos conseguir un programa lo más parecido posible al último, desde luego con nuestras propias ideas, pero procurando utilizar todos nuestros conocimientos. Podemos ejercitarnos a partir de ahora con otros programas parecidos, como por ejemplo, uno que halle el doble de un número, o incluso uno que realice la suma de dos números, aunque en ese caso deberemos utilizar dos **INPUT**, uno para cada uno de los sumandos. Ahí va una pista sobre este último programa propuesto:

```
10 INPUT A
20 INPUT B
30 PRINT A+B
```

DISEÑO DE PANTALLAS

TODOS sabemos, que la pantalla es el medio fundamental que el ordenador utiliza para comunicarse con nosotros. Por ello, a la hora de confeccionar buenos programas, hemos de poner gran empeño en hacer de la pantalla un agradable vehículo de información; esto se consigue cuidando la estética de la misma en sus diferentes aspectos:

— Diseño de la pantalla.
— Utilización del color.
— Dinamismo.

En los tres puntos anteriormente expuestos, se pueden concretar nuestras acciones sobre la pantalla para conseguir una presentación agradable de la misma. A continuación, vamos a dar repaso a una serie de trucos y consejos acerca de este asunto.

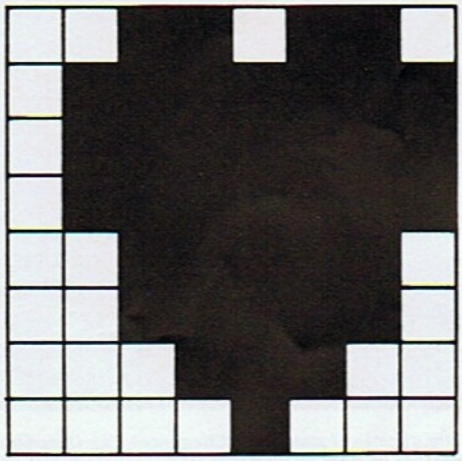


Gráfico definido de un corazón, que puede ser empleado para la confección de naipes, en cualquier juego basado en la baraja francesa.

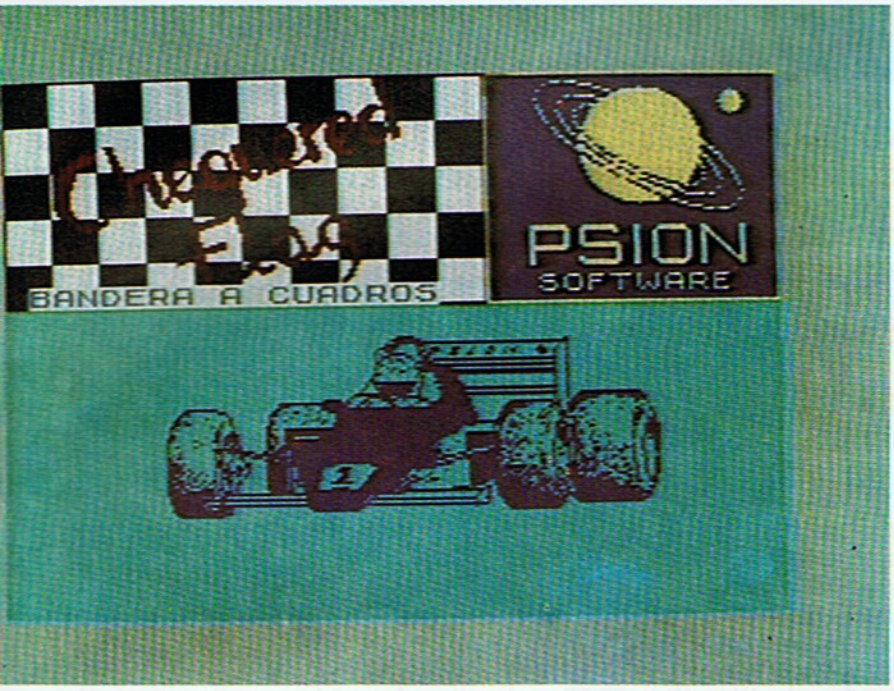
DISEÑO DE LA PANTALLA

Indudablemente, el diseño de la pantalla en sí es una parte esencial dentro de la estética de la misma, pero no tanto por la complejidad de las figuras que la integran, como por su distribución en el espacio reservado a la presentación visual. Durante estas primeras semanas de nuestra andadura por el BASIC del Spectrum, hemos tenido oportunidad de ver muchas figuras de considerable belleza, cuya realización no entraña por el contrario complejidad alguna, sino que son llevadas a cabo con tan solo una instrucción.

Llegados a este punto, debemos hacer una distinción entre dos aspectos fundamentales que contribuyen a un buen diseño de la pantalla: por una parte, las figuras que se presentan en ella y por otra, su distribución en la pantalla.

En el primer punto, influirán muy directamente los medios de que dispongamos para el diseño de las figuras y, desde luego, el tiempo que podamos invertir en su realización. En el mercado existen programas de utilidad, destinados precisamente a la realización de dibujos en la pantalla. Desde los modestos generadores de caracte-

ULTIMATE, PLAY THE GAME, *es una firma británica de reconocido prestigio, que destaca por la gran calidad de las pantallas de sus programas.*



Pantalla inicial del programa **Chequered flag** *(Bandera a cuadros) de* **PSION COMPUTERS.**

res, hasta los complejos programas conocidos como diseñadores de pantallas. Estos facilitarán enormemente nuestra labor estética, aunque sin ellos, pero con un mayor esfuerzo, también se pueden conseguir interesantes resultados.

Sin embargo, otro de los puntos importantes en la realización de un bonito trabajo, es la distribución inteligente de los «bultos» en la pantalla. Veamos esto mediante un ejemplo: supongamos que queremos realizar una pantalla de presentación de un juego basado en una baraja francesa. La primera fase del diseño es puramente imaginativa. En ella decidiremos cuál es la pantalla que deseamos realizar. Sin un gran esfuerzo, podemos llegar a la conclusión que una de las alusiones más claras a los juegos de este tipo, es el tapete de juego (verde) sobre el que se disponen los cuatro ases de la baraja.

Una vez que sabemos el tipo de diseño que vamos a realizar, debemos poner manos a la obra. El dibujo del tapete no será cosa complicada; bastará con asignar un código de color 4 al marco y al fondo de la pantalla, con lo que tras una instrucción **CLS** generaremos un perfecto tapete de juego. Llevémoslo a la práctica:

```
20 BORDER 4: PAPER 4: BRIGHT 0: CLS
```

Seguro que en la instrucción hemos observado un pequeño detalle: la desconexión del modo **BRIGHT**. De no ser así, y puesto que el brillo suplementario no afecta al marco de la pantalla, tendríamos un tapete verde claro por su zona interior, y verde oscuro en su periferia, cosa no muy común entre los paños de este tipo.

Un último toque podría aportarlo una línea roja de delimitación de los bordes del tapete. Aunque, según lo considere el programador, ésta puede tener un efecto negativo: reducir las dimensiones aparentes de la pantalla. Por tanto, vamos a dejar la línea de delimitación en una instrucción independiente, con el fin de poder suprimirla en el momento que nos interese.

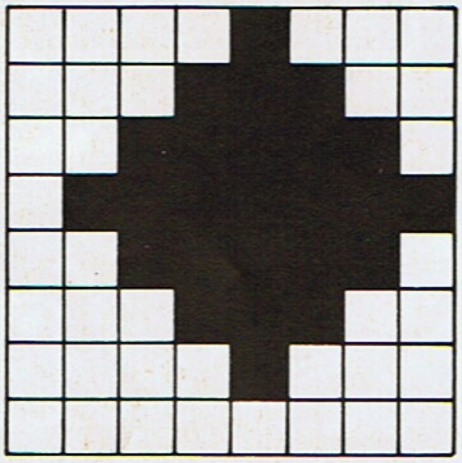
```
30 INK 2: PLOT 10,0: DRAW -10,10,PI/2:
DRAW 0,155: DRAW 10,10,PI/2: DRAW
235,0: DRAW 10,-10,PI/2: DRAW 0,-155:
DRAW -10,-10,PI/2: DRAW -235,0
40 PLOT 12,2: DRAW -10,10,PI/2: DRAW
0,151: DRAW 10,10,PI/2: DRAW 231,0:
DRAW 10,-10,PI/2: DRAW 0,-151: DRAW
-10,-10,PI/2: DRAW -231,0
```

La instrucción 40 traza otra línea de delimitación, un **pixel** más adentro que la dibujada por la 30.

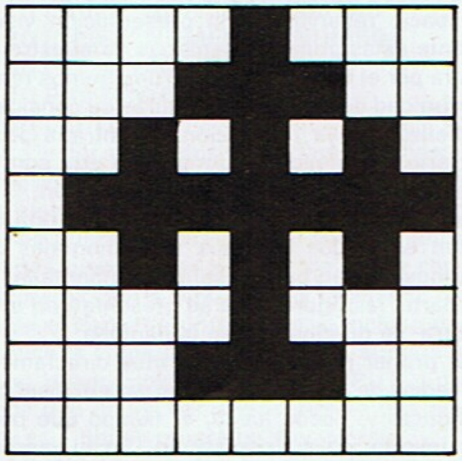
BITS

Una de las muchas aplicaciones de **OVER**, es conseguir la edición de mensajes parpadeantes, mejorando el efecto estético que se consigue mediante el modo **FLASH**. He aquí un ejemplo de ello.

```
10 PRINT AT
10,5;"MENSAJE PAR-
PADEANTE"
20 OVER 1: PAUSE
25: GO TO 10
```



Para conseguir la simetría en el gráfico del diamante, ha sido necesario dejar libres la primera columna y la última fila.



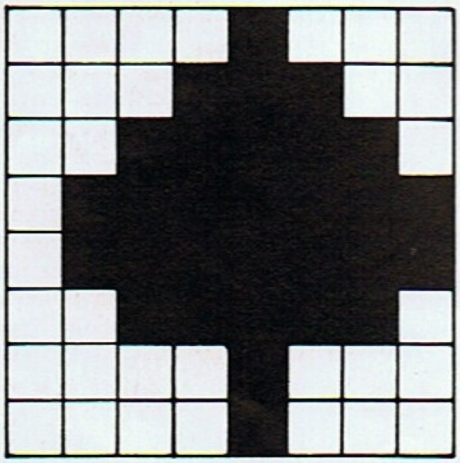
De toda la baraja, el gráfico más complicado de diseñar es el trébol, debido al detalle necesario para sus hojas.

La eliminación de ambas instrucciones producirá la supresión de la doble raya del tapete, sin afectar a ninguna otra instrucción posterior.

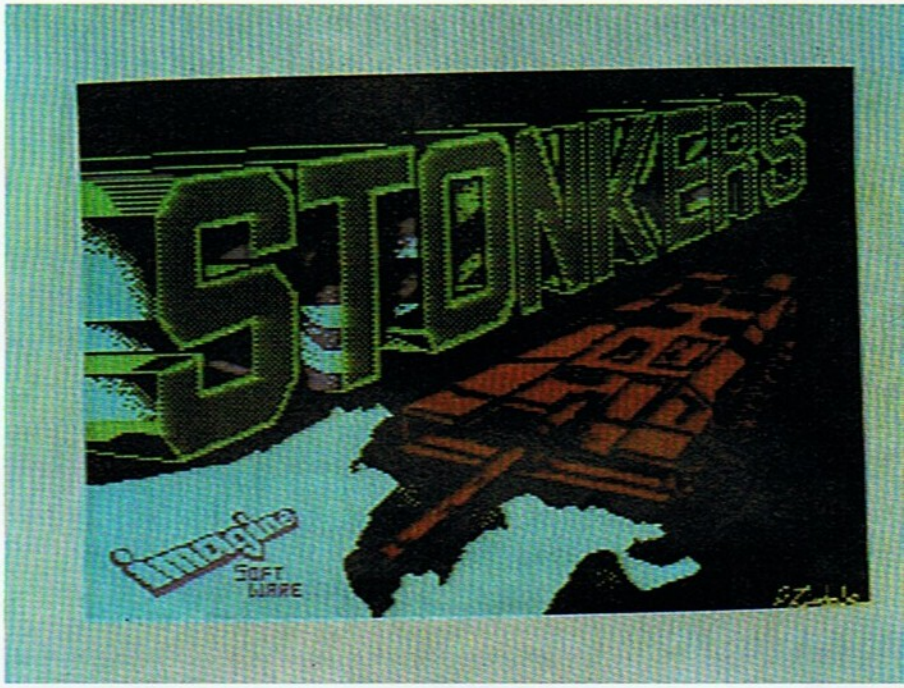
LOS NAIPES

Por fin, ha llegado la hora de perfilar el motivo central de la pantalla. Como en el caso del tapete, la realización del fondo blanco de la carta no supone mayores dificultades. Sin embargo, otro asunto es la realización del dibujo interior y la adecuada proporción de los naipes. La política más acertada, en este caso sería emplear para la pantalla los mismos ases que se utilizarán más tarde en el juego, con lo que nos ahorraremos una buena cantidad de trabajo. Las dimensiones de las cartas elegidas para nuestro ejemplo son 9 de alto y 7 de ancho, expresadas en posiciones de carácter (9 filas × 7 columnas).
Lo mejor sería, desde luego, plasmar el fondo de las cartas mediante subrutina, pues será una tarea que habrá que realizar muy frecuentemente en el juego. Además, la posición donde se debe efectuar la impresión variará, por lo que ha de ser indicada a la subrutina mediante unos parámetros; por ejemplo, F (fila) y C (columna).

```
999 STOP: REM Fin de programa principal
1000 FOR I=0 TO 8: PRINT AT F+I,C; PAPER 7; "       ": NEXT I
1020 RETURN
```



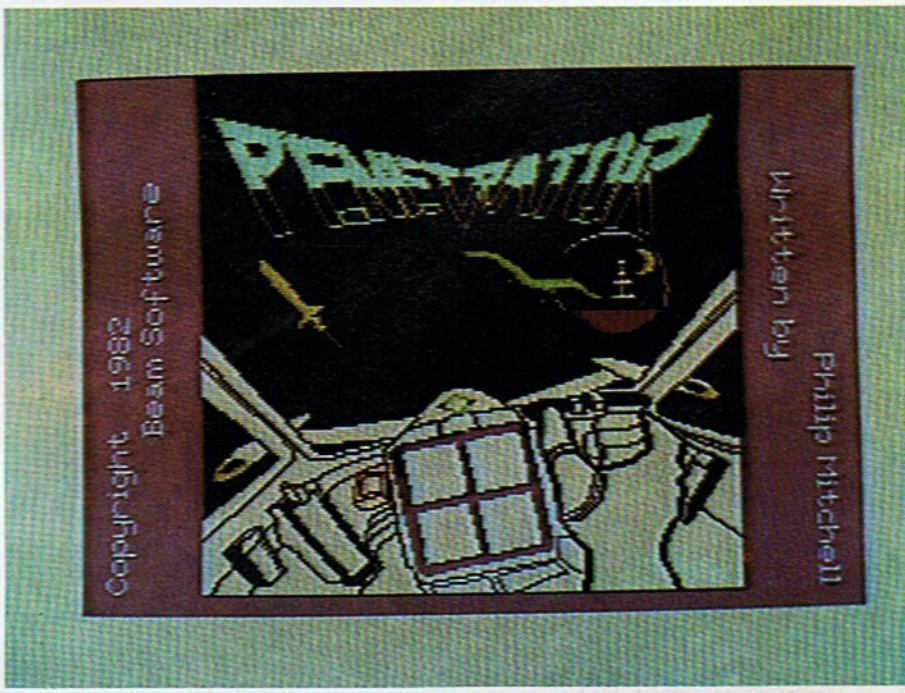
La pica es el último de los gráficos definidos para la baraja, con unas características muy similares al corazón en cuanto a dimensiones.



Pantalla de presentación del programa de estrategia militar **Stonkers.**

Antes de pasar al dibujo de las cartas, vamos a completar la subrutina que dibuja el naipe con un detalle más: un marco que evite la difuminación del blanco sobre el fondo verde del tapete. Para ello utilizaremos una línea muy fina, del color de la carta; es decir, roja para corazones y dia-

Penetrator, *una versión para el Spectrum del popular* **Scramble.**



Pantalla inicial del programa Atic-Atac.

mantes, y negra para tréboles y picas. Evidentemente, se nos hace necesario un tercer parámetro, que indique a la subrutina el color de la carta. Utilicemos por ejemplo la variable CC, que contendrá el código de color de la carta (0=negro y 2=rojo).

```
1020 LET FP=F*8
1030 LET CP=175-C*8
1040 INK CC: PLOT FP,CP
1050 DRAW 56,0: DRAW 0,-72: DRAW -56,0: DRAW 0,72
1060 RETURN
```

El siguiente paso es, sin duda alguna, la definición de unos caracteres gráficos, que representen cada uno de los palos de la baraja. Esto será trabajo fácil, ya que la cinta de demostración que acompaña al Spectrum contiene un programa generador de caracteres, que nos ayudará en esta tarea. Por tanto, en el programa principal incluiremos la definición de los caracteres, que en nuestro ejemplo se han localizado en las letras A, B, C y D, para corazón, diamante, trébol y pica, respectivamente.

```
50 FOR I=0 TO 31: READ A: POKE USR "A"+I,A: NEXT I
60 DATA 54,127,127,127,62,62,28,8
70 DATA 0,8,28,62,127,62,28,8
80 DATA 8,28,42,127,42,8,28,0
90 DATA 8,28,62,127,127,62,8,8
```

Nosotros también podemos diseñar nuestras propias pantallas. He aquí una muestra de ello.

Para evitar problemas de color, se pueden difuminar los contornos de las figuras.

Para los que no dispongamos de más facilidades que el programa generador de caracteres, la primera parte de la labor de diseño ya ha terminado. Sólo es necesario situar el correspondiente gráfico en el centro de la carta y en las esquinas superior izquierda e inferior derecha de la misma. Por el contrario, los más afortunados en cuanto a los medios de diseño gráfico, podremos acometer el dibujo de unos ases de grandes dimensiones. El sistema de codificación de los mismos variará a gusto del programador, pero una manera bastante cómoda de almacenar la forma de los ases puede ser unas sentencias **DATA**, que contengan direcciones de **PLOT** y coordenadas de **DRAW**.

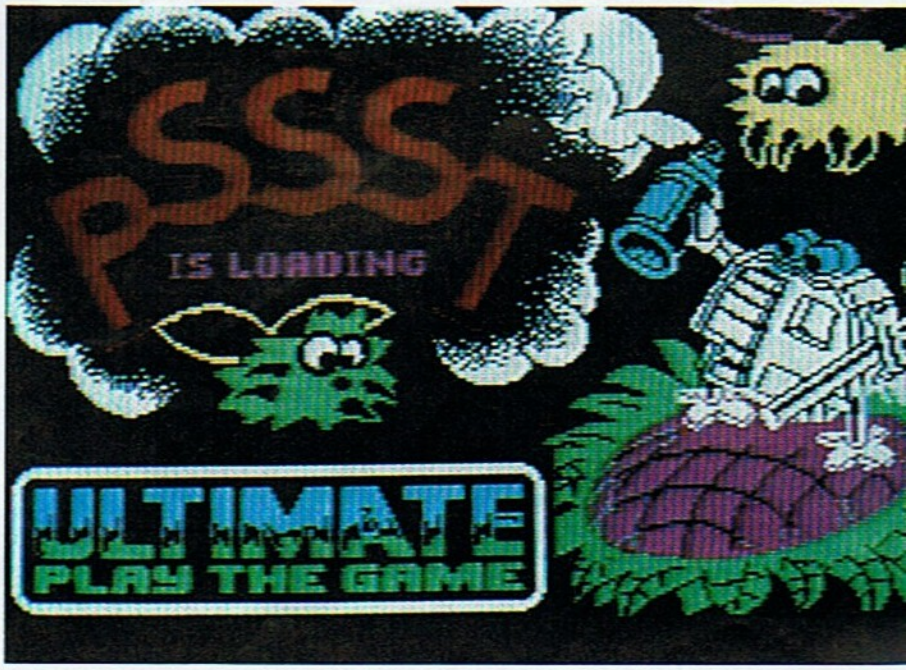
Con todo lo visto hasta el momento, seremos capaces de diseñar nuestra propia pantalla. Sólo tenemos que ensamblar adecuadamente las diferentes partes que hemos ido estudiando. Por supuesto, las rutinas aquí expuestas son meras sugerencias, que pueden ser complementadas y mejoradas a gusto del programador; por ejemplo, buscando sistemas de dinamismo en la pantalla, para que ésta no resulte tan estática. Una idea en este sentido, es establecer la entrada de mensajes con el título del programa, nuestro nombre o las instrucciones.

ULTIMOS CONSEJOS

Para terminar, vamos a dar un pequeño repaso, a todos los aspectos importantes que hemos visto para el diseño de una pantalla.

* Es muy importante que la pantalla sea representativa de la idea que se desea comunicar.
* Debemos procurar, en lo posible, enmarcar con una línea fina las masas de color superpuestas sobre fondos distintos. Cuanto más pequeña sea la zona a enmarcar, más necesaria será esta medida. Sin llegar, por supuesto, al extremo de delimitar zonas de un solo carácter.
* La delimitación mediante línea del marco de la pantalla puede ser un arma de doble filo, ya que inevitablemente reduce las dimensiones aparentes de la pantalla.
* Es fundamental la distribución de las figuras. Estas deben guardar simetría consigo mismas y con la pantalla. Es necesario eliminar, en lo posible, los espacios vacíos respecto a espacios excesivamente cargados.

Bien, esperamos que estos consejos hayan sido lo suficientemente aleccionadores, como para convertirnos en unos grandes diseñadores de pantallas. El tema aún nos reserva algunas agradables sorpresas, como por ejemplo, cuando abordemos los espectaculares efectos de dinamismo de la pantalla, que se pueden conseguir con unos pocos trucos.



La combinación de colores en la pantalla, requiere bastante experiencia para poder obtener resultados como el de la foto.

SATURNO JET

COMO piloto más experimentado de la Armada Interplanetaria, se te ha encomendado la difícil misión de tripular la nave Saturno Jet, en misión de reconocimiento por los anillos del planeta Saturno.

Las últimas noticias llegadas de la base Lambda 4 en el satélite Titán, son bien poco alentadoras: los malvados habitantes del planeta Burko (según se sale de Tau Ceti, el tercer planeta a la izquierda), han enviado su flota de conquista a nuestro sistema solar.

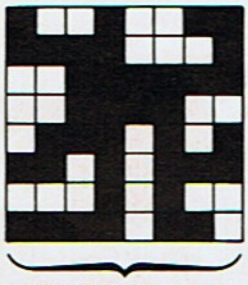
La única manera de hacerles frente con ciertas posibilidades de victoria, es conocer su poderío concreto; por ello, han distribuido sus naves en la zona de los anillos de Saturno, haciéndolas invisibles a los medios habituales de detección terrestres. Tu objetivo es tripular por control remoto la nave de reconocimiento Saturno Jet, introduciéndote en las líneas burkonianas, para informar de la composición de su escuadra.

Cuanto más te adentres en los anillos, tanto más difícil será tu misión, pero más importante será la información que los ojos electrónicos de tu nave transmitan. No dudes que para evitar tu acción de espionaje, los pérfidos burkonianos utilizarán todos los medios a su alcance.

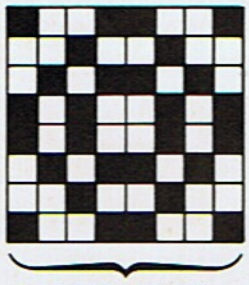
Iniciarán sus operaciones disuasorias mediante el lanzamiento de Burkominas y, si consigues esquivarlas, pondrán en juego sus Star Burko Fortress (naves burkonianas de invasión masificada), en un último esfuerzo por conseguir lo que las Burkominas no lograron llevar a buen término.

Recuerda siempre que tu nave es de reconocimiento y, como tal, no dispone de ningún mecanismo de ataque o defensa, sino simplemente de una gran maniobrabilidad. Deberás evitar siempre el choque con cualquiera de las naves burkonianas, o su zona de influencia (una columna de pantalla intergaláctica a cada lado), ya que ésto supondría la inmediata destrucción de la nave que tripulas.

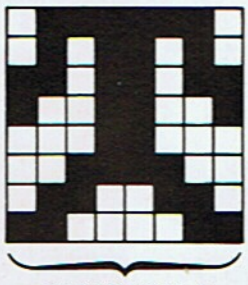
Para el control del Saturno Jet, emplearás las teclas **Z** (izquierda) y **X** (derecha), presentes en el panel de control de tu ordenador, en el Cuartel General Terrestre.



BURKOMINA 1
GRAF. EN «C»

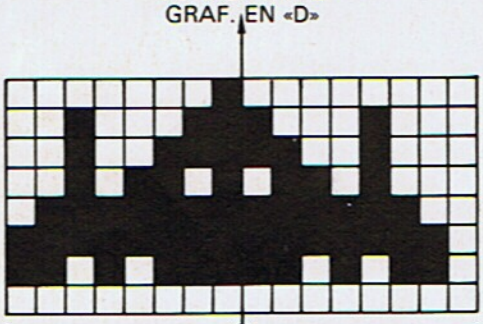


BURKOMINA 2
GRAF. EN «D»

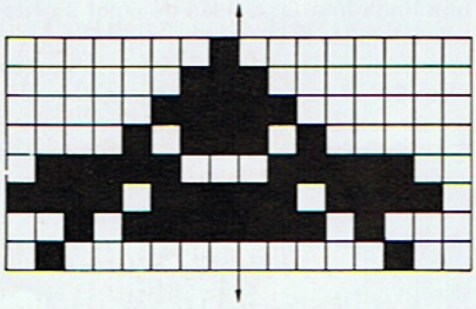


BURKOMINA 3
GRAF. EN «E»

La forma que adoptan las naves burkonianas y el Saturno Jet, se debe a los gráficos definidos indicados en el dibujo. Su modificación es muy sencilla de incluir al comienzo del programa.

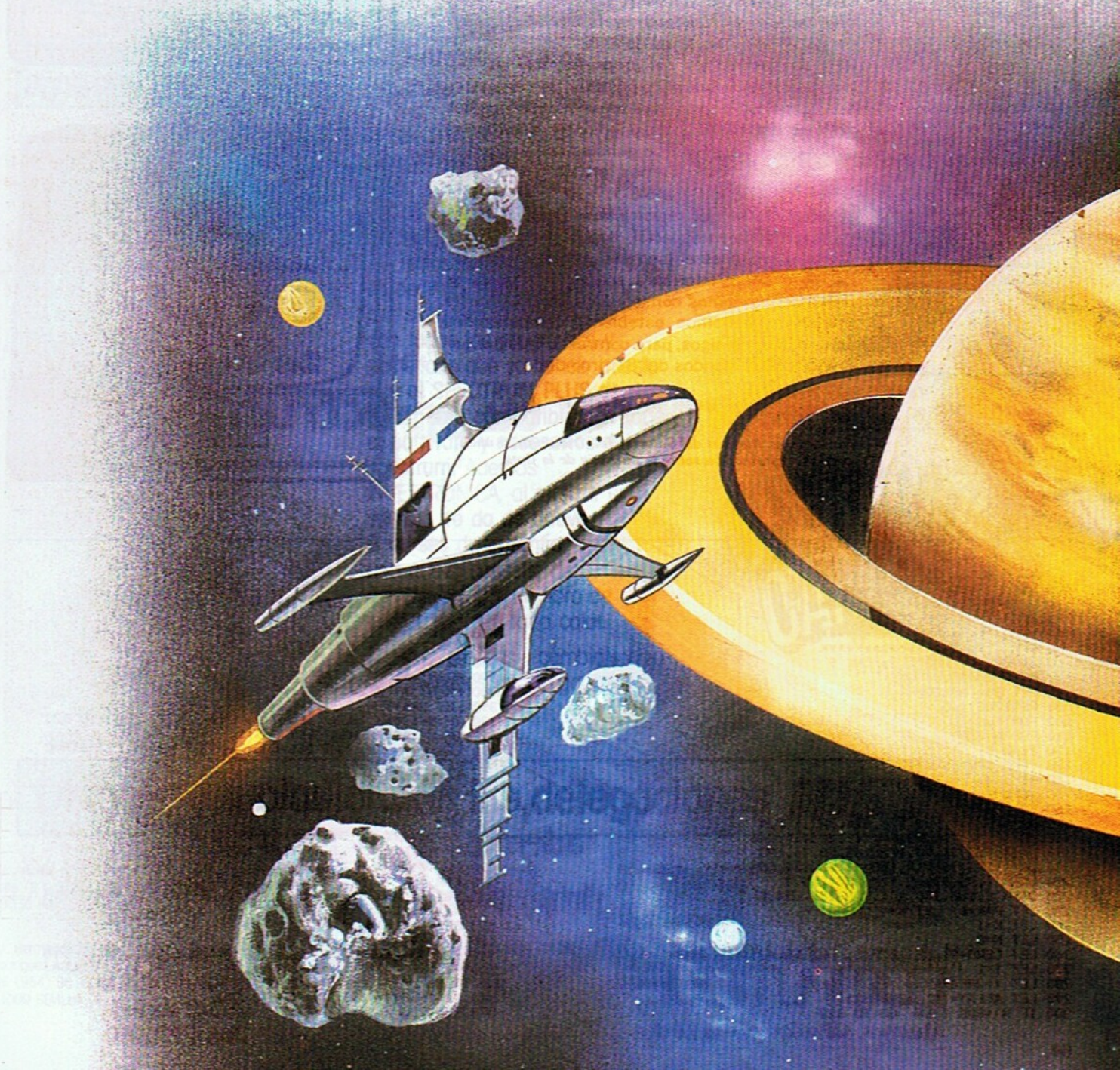


GRAF. EN «F» GRAF. EN «G»
STAR BURKO FORTRESS



GRAF. EN «A» GRAF. EN «B»
JUPITER JET

El programa se basa en la utilización de la variable del sistema SCR CT **(SCRoll CounTer)**, localizada en la dirección decimal 23692. La misión de esta variable, es contar el número de líneas desplazadas fuera de la pantalla, desde el último mensaje de **Scroll?** emitido. Evitando mediante POKEs que alcance el valor cero, conseguiremos que las oleadas de «estorbos espaciales» no se



BITS

El control de la variable del sistema SCR CT, *permite evitar la aparición del desagradable mensaje* Scroll?, *en plena batalla.*

Al introducir el programa, no olvidemos sustituir los caracteres subrayados, por los gráficos definidos correspondientes a las teclas indicadas.

*

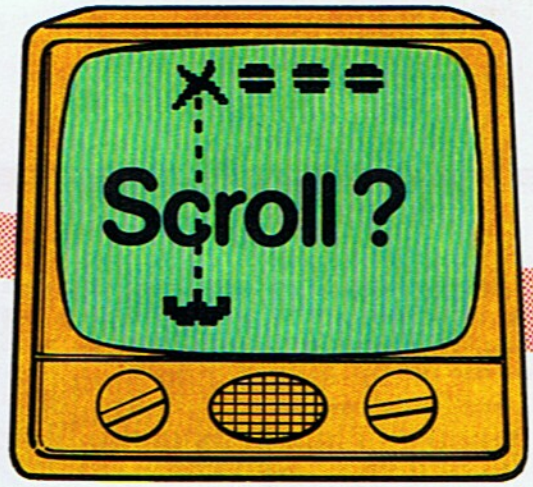
Para el movimiento del Saturno Jet, emplearemos las teclas **Z** (izquierda) y **X** (derecha). Tengamos siempre bien presente, que debemos evitar la colisión con las naves burkonianas, puesto que no disponemos de dispositivos de defensa y ataque.

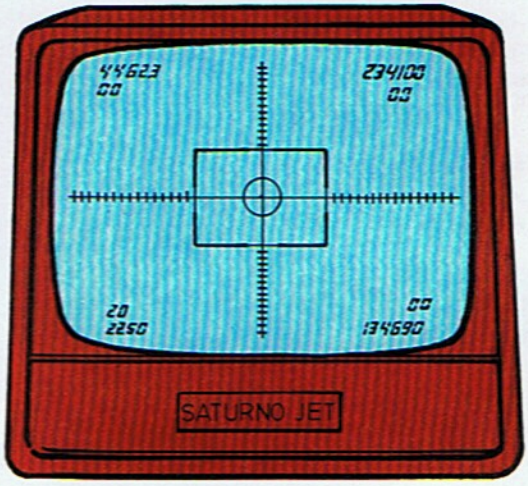
detengan en ningún momento, con el mensaje de desplazamiento de la pantalla.

La forma de simular el movimiento de las naves burkonianas ha sido bien simple: todo se ha limitado a escribir por cada ciclo de programa, una nave enemiga en un punto aleatorio de la última línea de la pantalla; al forzarse el carácter **RETURN** después de la escritura de cada nave, se efectúa un desplazamiento automático de la pantalla **(scroll)**, por haberse realizado la representación en la última línea de ésta.

Para la introducción del programa, hay que seguir la norma habitual de sustituir los caracteres que aparecen subrayados en el listado, por los gráficos definibles correspondientes a las letras subrayadas. Una vez que el listado se encuentre en la memoria, podemos grabarlo mediante **SAVE "SATURNO" LINE 1.**

El momento en que nuestra nave pase a convertirse en chatarra espacial, llegará tarde o temprano, y entonces el juego se dará por terminado; pero antes informará sobre cuántas naves burkonianas hemos conseguido evitar. Con este marcador, podemos establecer competiciones con nuestros amigos, para contrastar nuestra pericia ante los mandos del Saturno Jet.





Los ojos electrónicos del Saturno Jet, transmitirán una preciada información sobre la potencia de la flota burkoniana.

```
 10 REM   ******************
 20 REM   ******************
 30 REM   **  J.M.MAYORAL  **
 40 REM   ******************
 50 REM   ******************
 60 BORDER 0: PAPER 0: INK 9
 70 POKE 23658,8: CLS
 80 LET a$='ABCDEFG'
 90 RESTORE 160
100 FOR K=1 TO 7
110 FOR N=0 TO 7
120 READ P
130 POKE USR A$(K)+N,P
140 NEXT N
150 NEXT K
160 DATA 1,3,7,27,124,247,47,64
170 DATA 0,128,192,176,124,222,232,4
180 DATA 147,241,63,60,117,215,20,246
190 DATA 153,36,90,165,165,90,36,153
200 DATA 126,90,219,153,24,60,102,195
210 DATA 1,35,39,45,127,255,215,0
220 DATA 0,136,200,104,252,254,214,0
230 LET PAR=6: LET A$='CDEFG'
240 LET K=1
250 LET N=0
260 LET CERO=0: LET BTM=0: LET COL=0: LET DES=0
270 LET T=1
280 LET X=2*PAR
290 LET ALEAT=INT (RND*32)
300 IF N>=300 THEN  GO TO 320
310 LET B$=A$(K)
320 PRINT AT 21,ALEAT; INK INT (RND*7)+1;B$
330 POKE 23692,255
340 PRINT AT 21,31;'  '
350 REM
360 PRINT AT 10,X-2;'      '
370 PRINT AT 11,X;'AB'
380 PRINT AT 21,31;'  '
390 LET N=N+T
400 IF N=100*3 THEN  LET T=2: LET B$='FG': GO TO 420
410 IF N=100*K THEN  LET K=K+1
420 LET PUM=DES: LET DES=COL: LET COL=BTM: LET BTM=C
ERO: LET CERO=ALEAT
430 PRINT AT 10,X-2;'      '
440 PRINT AT 11,X;'AB'
450 IF X>=PUM-2 AND X<=PUM+T THEN  GO TO 500
460 REM
470 IF INKEY$='Z' THEN  LET X=X-T
480 IF INKEY$='X' THEN  LET X=X+T
490 GO TO 290
500 FOR V=1 TO 5
510 PRINT  INK INT (RND*7)+1; OVER 1;AT 11,X;A$(V);A
T 11,X+1;A$(V)
520 BEEP .1,-10
530 NEXT V
540 PRINT  PAPER 7;AT 21,0;' PUNTUACION='; PAPER 6;N
550 PRINT  PAPER 6; FLASH 1; BRIGHT 1;AT 0,3;'PULSA
UNA TECLA PARA JUGAR'
560 IF INKEY$='' THEN  GO TO 560
570 RUN 230
```

MANEJO DEL CASSETTE

NO resulta nada práctico tener que teclear los programas cada vez que deseamos ejecutarlos de nuevo, sobre todo si contienen un número grande de instrucciones. La solución a estos problemas es el empleo del casete.

El Spectrum ofrece la posibilidad de utilizar un casete convencional para almacenar programas y datos. Podemos decir que para este menester sirve cualquier aparato de este tipo, pero son preferibles los equipos **mono** de bajo coste, a los **estéreo.**

Lo que también resulta deseable es que el casete disponga de cuentavueltas, así como de un sistema de avance y retroceso rápido de la cinta, que permita la salida por el altavoz mientras se realizan estas operaciones. En todo caso, estas no son cualidades imprescindibles.

Por el contrario, es absolutamente esencial que el casete disponga de una entrada de micrófono **MIC,** y una salida para cascos **EAR,** en forma de hembras de 3,5 mm. Si el equipo no tiene este tipo de conexiones, debemos adquirir las correspondientes clavijas de adaptación en una tienda especializada. Sin embargo, esto no será necesario con la mayoría de los aparatos modernos, fundamentalmente de marcas japonesas, ya que la práctica totalidad de los mismos incorporan este tipo de conexiones.

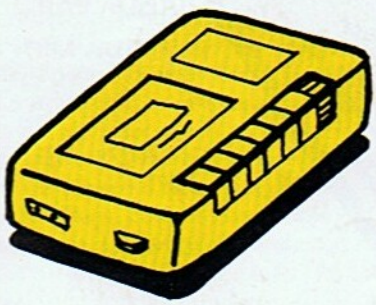
En la parte trasera del Spectrum, entre los conectores de antena y red, existen dos conexiones marcadas **MIC** y **EAR** de 3,5 mm., que se corresponden con las entradas del mismo nombre en el casete.

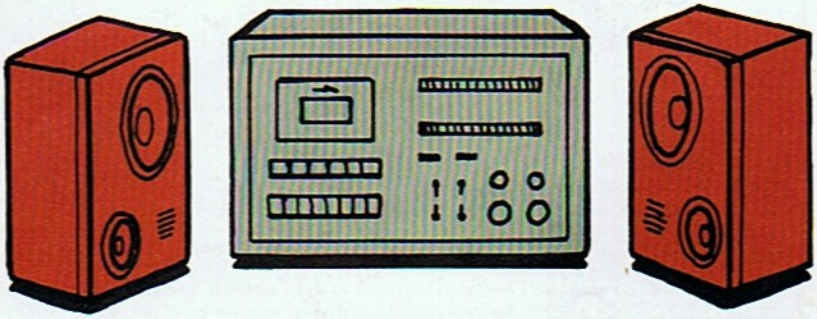
Junto con el ordenador se suministra un cable doble, con dos clavijas en cada uno de sus extremos, unas de color negro y otras de color gris, que sirven para comunicar el ordenador directamente con el casete; esta operación debe ser realizada entre las entradas y salidas homólogas, es decir, **MIC** con **MIC** y **EAR** con **EAR.** Como es lógico, no importa qué color de clavija se asigne a cada una de estas conexiones, pero sí es interesante que se tome una determinación para que, en adelante, utilicemos el mismo color para cada conexión en concreto. Podemos utilizar, por ejemplo, las clavijas de color gris para la conexión de **EAR** con **EAR,** y las de color negro para **MIC** con **MIC.**

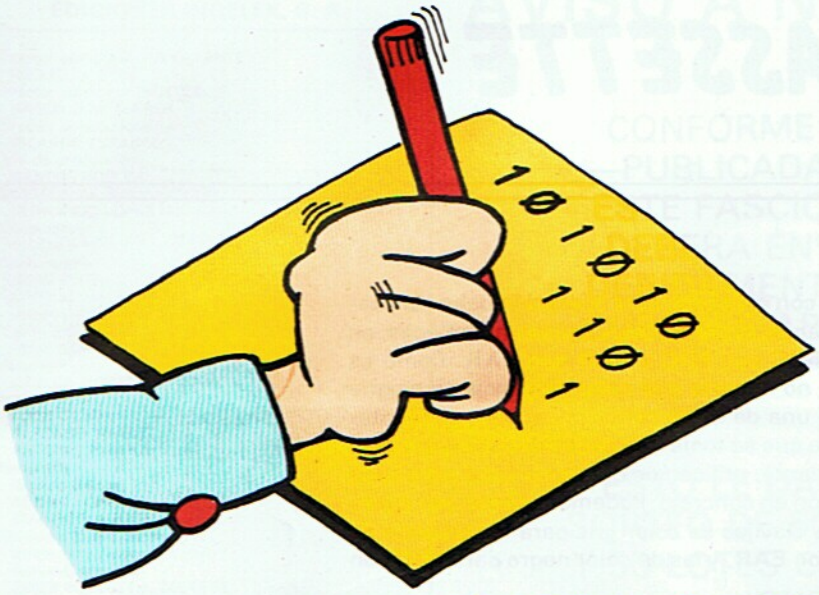
Cuando deseemos almacenar el programa de la memoria en la cinta, debemos emplear las conexiones del tipo **MIC,** en nuestro ejemplo las cla-

Los usuarios que poseemos un casete, nos evitamos el trabajo de tener que teclear de nuevo los programas al encender el ordenador.

Para la grabación y lectura de programas, la utilización de equipos mono ***de bajo coste, suele ser preferible a la de los caros aparatos*** estéreo.







La sentencia SAVE *facilita la «escritura» de programas.*

BASIC PARA EL CASETE

El Spectrum dispone de tres palabras clave para el manejo del casete, que se corresponden con las funciones de grabación, verificación y lectura de programas, y van siempre acompañadas del nombre del programa entre comillas ("). Estas palabras son **SAVE**, en inglés salvar, **VERIFY**, en inglés verificar y **LOAD**, en inglés cargar, respectivamente.

Vamos a suponer ahora que aún permanece en la memoria nuestro último programa, y deseamos almacenarlo en la cinta con el nombre de CUADRADOS. Si no lo tenemos, porque desconectamos el ordenador al terminar de modificarlo, debemos teclearlo de nuevo. Pero esto no debe molestarnos, porque el primer paso para llegar a dominar realmente nuestro Spectrum, es adquirir práctica en el manejo del teclado. Recordemos que el programa era:

```
 5 PRINT "NUMERO","CUADRADO"
10 INPUT "Numero:";N
20 LET C=N*N
30 PRINT N,C
40 GO TO 10
```

Está claro que debemos emplear la instrucción **SAVE**, que es la que se utiliza para grabar programas en cinta, pero antes debemos realizar al-

vijas negras, desconectando cualquiera de las grises (es indiferente la del casete o la del ordenador) para evitar problemas de interferencias en la grabación. Del mismo modo, cuando acabamos de encender el ordenador, y lo que queremos es recuperar un programa grabado anteriormente en la cinta, debemos emplear las conexiones de color gris **(EAR)**, desconectando alguna de las de color negro, de forma similar al caso anterior. Es importante recordar el hecho de que, a la hora de almacenar un programa en casete, las clavijas de tipo **MIC** deben estar conectadas, mientras que alguna de las de **EAR**, o las dos, no deben estarlo.

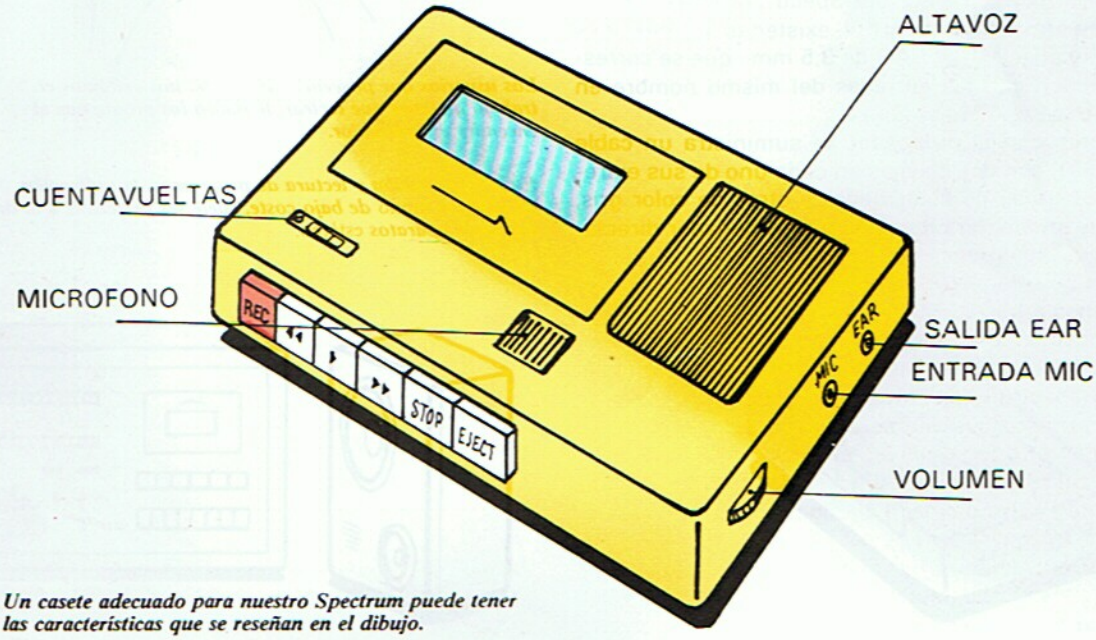
Un nombre de programa está formado por una secuencia de 1 a 10 caracteres cualesquiera, siendo obligatorio especificarlo como parámetro de la sentencia **SAVE**, si bien no para **VERIFY** y **LOAD**.

*

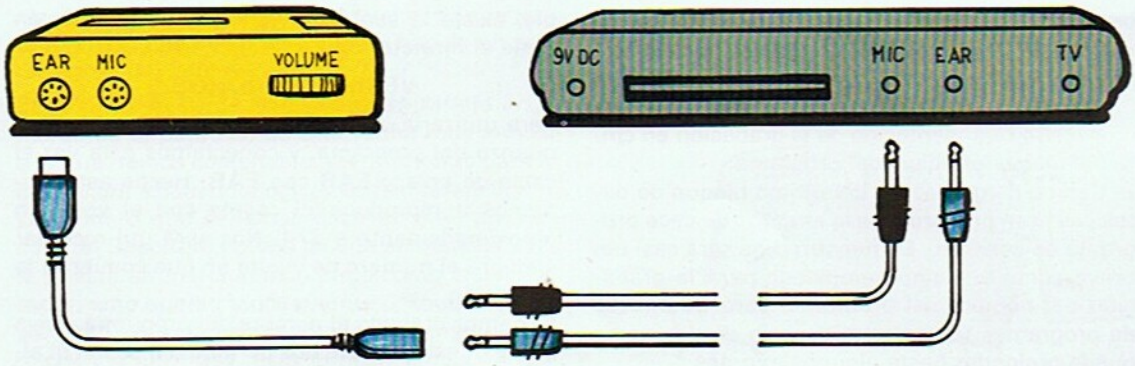
La conexión entre casete y ordenador, se efectúa a través de las salidas marcadas como **MIC** y **EAR** en ambos aparatos.

*

No es preciso que el casete que utilicemos con el Spectrum esté dotado de una entrada para control remoto **(REMOTE)**.



Un casete adecuado para nuestro Spectrum puede tener las características que se reseñan en el dibujo.



Existen adaptadores de clavijas que facilitan el uso de casetes que no posean conectores de 3,5 mm.

gunas operaciones de preparación de la grabación.

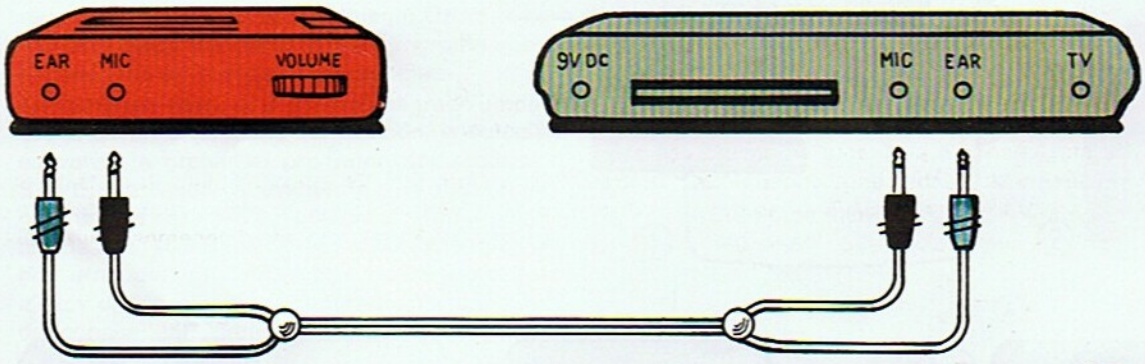
La primera es avanzar la cinta hasta un sitio donde podamos grabar. Si se trata del comienzo de la misma, debemos sobrepasar la zona de cinta plástica de principio. Aunque no es obligatorio, sí es muy conveniente que antes de almacenar un programa en cinta, grabemos de viva voz el nombre del mismo y lo anotemos en la carpetilla del casete junto con el número de vuelta en que se encuentra. Hecho esto, conectamos las tomas de **MIC** del ordenador y del casete a través del cable al efecto, manteniendo sin conectar una de las clavijas de **EAR** (o las dos). Una vez realizada esta preparación, podemos pedir a nuestro ordenador que lleve a cabo la grabación:

SAVE "CUADRADOS"

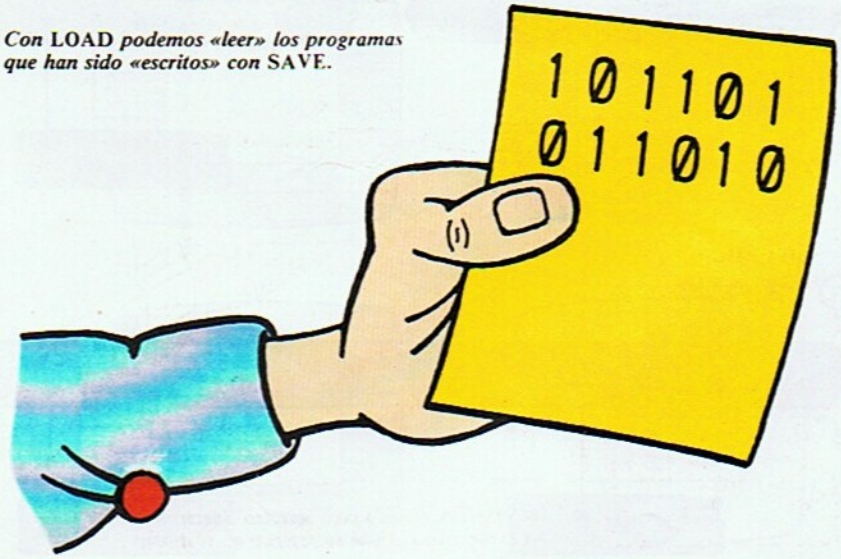
Pulsamos **ENTER** y aparece el mensaje **Start tape, then press any key** (conecta el casete y pulsa cualquier tecla). Y es exactamente eso lo que vamos a hacer. Colocamos el casete en modo de grabación (generalmente pulsando RECORD y PLAY) y, acto seguido, pulsamos cualquier tecla del Spectrum (distinta de los **SHIFT**s).

Lo primero que vemos aparecer en el marco de la pantalla son unas bandas anchas de color rojo y azul claro, que se desplazan lentamente, durante cuatro o cinco segundos, y a continuación una ráfaga casi imperceptible, de menos de un segundo de duración, de bandas de color azul oscuro y amarillo, mucho más estrechas que las anteriores, con las que se graba en la cinta el nombre del programa y algunos otros datos que el ordenador necesitará más tarde para identificarlo. Esta información se denomina *HEADER* (aproximadamente se pronuncia *hider)* o CABECERA. Tras un cortísimo espacio de tiempo, otra vez las

Las clavijas de igual color habrán de unir las conexiones que en el Spectrum y el casete tienen el mismo nombre.



Con LOAD *podemos «leer» los programas que han sido «escritos» con* SAVE.



BITS

La utilización con el ordenador de casetes **estéreo** no suele dar buenos resultados; por el contrario, encontraremos la máxima fiabilidad con los aparatos **mono**, de más bajo coste que los anteriores.

*

Aunque el Spectrum dispone de dos salidas, destinadas a las operaciones de grabación **(SAVE)** y lectura **(LOAD** y **VERIFY)**, podemos realizar ambas tareas por cualquiera de ellas.

*

Recordemos que en el momento de realizar una grabación, no deben estar conectadas las dos salidas **EAR** de casete y ordenador.

bandas anchas (azules y rojas) indicativas de comienzo de bloque de datos, esta vez por espacio de unos dos segundos y, por último, nuevamente las bandas finas de color azul oscuro y amarillo, que son la representación de la grabación en cinta del bloque principal del programa.

El tiempo de grabación del último bloque de datos, varía en proporción a la longitud de cada programa en concreto. En nuestro caso será casi tan breve como el tiempo empleado para la grabación del nombre del programa; pero tratándose de programas largos, este espacio de tiempo se puede prolongar hasta algunos minutos.

El mensaje **OK** nos indica que la grabación ha terminado correctamente, al menos en cuanto al Spectrum se refiere, porque el ordenador al grabar, no detecta si existe alguna deficiencia en la superficie de la cinta que está utilizando. Para evitar el riesgo de almacenar una copia de nuestro programa que pueda ser después irrecuperable, existe la sentencia **VERIFY.** La instrucción tiene el formato:

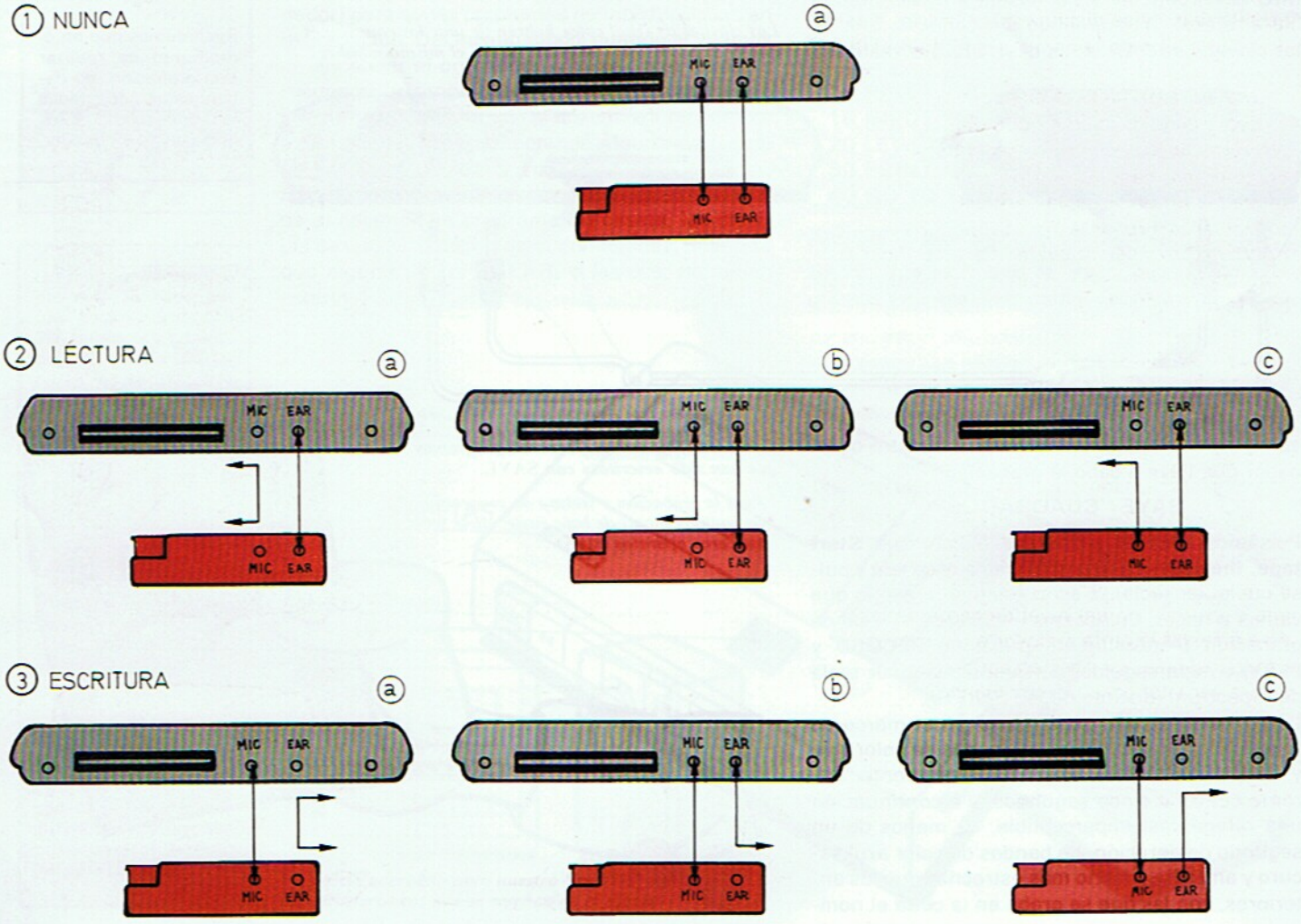
VERIFY "CUADROS"

Para utilizarla, rebobinamos la cinta hasta el comienzo del programa, y conectamos esta vez el cable de enlace **EAR** con **EAR.** Hecho esto, ponemos a reproducir el casete con el volumen aproximadamente a 3/4. Nos será útil recordar siempre el número de vuelta en que comienza la grabación.

Si hemos grabado el nombre del programa a viva voz, y nuestro casete nos permite dar salida al altavoz mientras rebobinamos la cinta, el comienzo se localizará muy fácilmente, ya que, frente al monótono ruido producido por los pitidos que ha grabado el Spectrum, destacará el sonido característico de una voz al ser reproducida a alta velocidad.

Para el Spectrum, la verificación consiste en leer el programa almacenado anteriormente en la cinta y compararlo con el que se encuentra en la memoria. Si los dos programas le resultan idénticos en longitud y contenido, responderá al final de la verificación con el mensaje **OK.** En caso contra-

Sistema de conexión entre casete y ordenador.



rio, se detendrá mostrando el mensaje **Tape loading error,** que nos indica que ha habido un problema en la verificación.

El proceso de verificación tiene su reflejo en el marco de la pantalla, de una forma muy similar a la que hemos comentado anteriormente para la grabación, existiendo lógicamente diferencias.

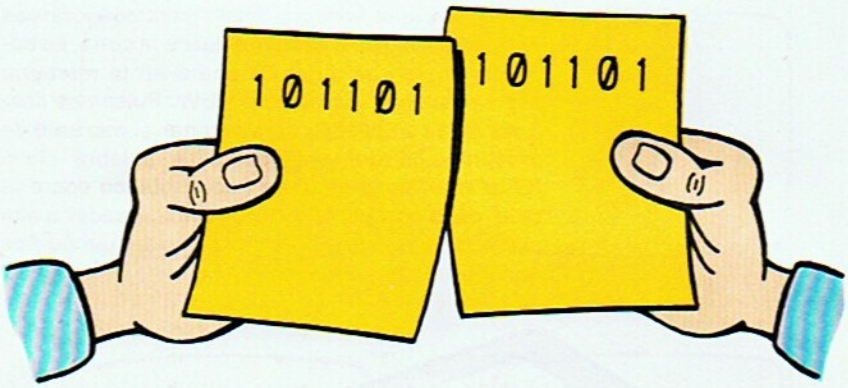
Lo primero que debemos decir es que, en este caso, el Spectrum no nos facilita el mensaje para la puesta en marcha de la cinta como la vez anterior, sino que inmediatamente después de pulsar **ENTER,** se coloca a la espera de recibir datos desde el casete. Esto lo observamos porque al conectar la clavija al casete, el Spectrum «reconoce» su conexión con él, y cambia los colores del marco de la pantalla del rojo al azul claro, de manera intermitente, para indicar esta circunstancia.

Seguidamente, aparecen las bandas anchas de color azul claro y rojo de comienzo de bloque de datos, y a continuación del *header* aparece en la pantalla el mensaje **Program: CUADRADOS,** indicativo de que el Spectrum ha identificado el nombre del programa grabado en la cinta. A continuación, vuelven a aparecer las bandas anchas hasta que el ordenador localiza el siguiente bloque de datos y se dispone a verificarlo, mostrando en el marco de la pantalla las bandas finas de colores azul oscuro y amarillo. Finalmente, aparecerá en la pantalla el mensaje de **OK,** que sirve como confirmación de que el programa quedó almacenado en la cinta correctamente.

En cualquier caso, si el proceso se interrumpe a causa de un error en la verificación, la solución es volver a grabar el programa por medio de **SAVE,** y proceder nuevamente a verificar hasta obtener los resultados apetecidos. Nadie nos libra de que, sólo de vez en cuando, tengamos algún error de verificación. Esto puede ser debido a muy diversas causas, generalmente derivadas del consiguiente deterioro de la cinta al ser borrada y vuelta a grabar en muchas ocasiones. Otras veces, el problema se debe a algún pequeño fallo en la alimentación del casete cuando éste está conectado a un adaptador de corriente alterna (sin pilas). A pesar de ello, son muchos más los problemas que pueden derivarse de realizar los procesos de grabación y recuperación de programas empleando las baterías del casete, ya que dependiendo del nivel de carga de las pilas, éste arrastra la cinta a mayor o menor velocidad. Esta diferencia, apenas perceptible para nosotros, puede ser vital para nuestro Spectrum.

De hecho, puede suceder que no se pueda cargar un programa grabado ya hace días, debido a que el estado actual de las baterías no coincide con el que se dio al efectuarse la grabación del programa. Afortunadamente, esto no suele suceder más que cuando la diferencia de velocidades es bastante notoria.

Vamos a practicar ahora con la instrucción

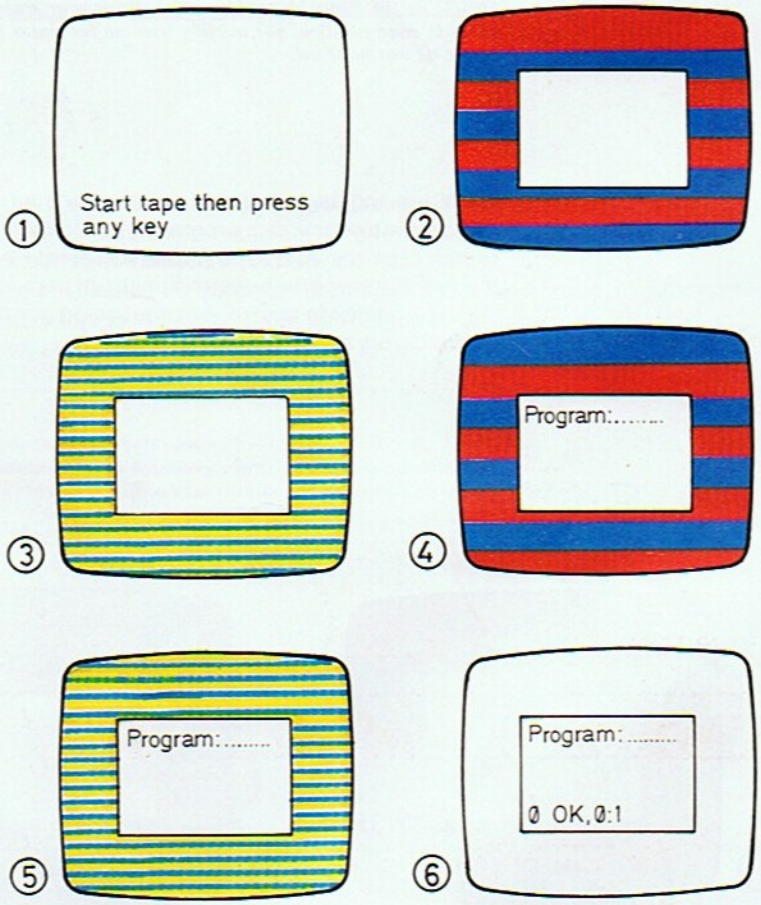


VERIFY *comprueba que los datos «escritos» por* **SAVE** *coinciden con los existentes en la memoria.*

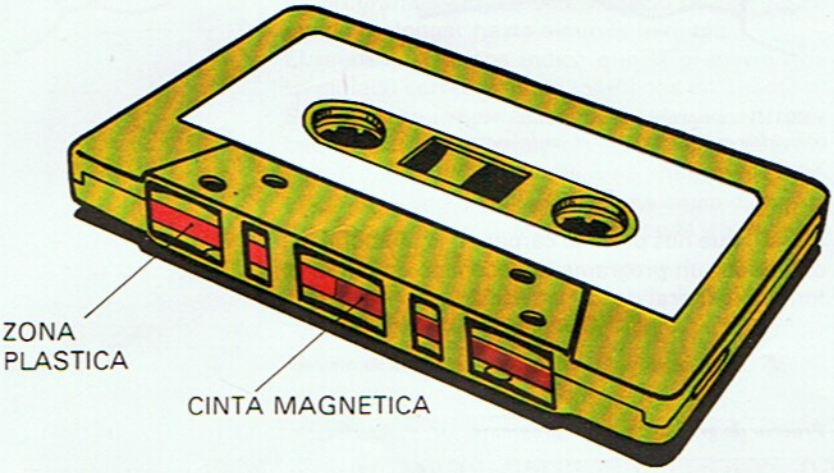
LOAD, que nos permite cargar en la memoria del ordenador un programa almacenado en cinta. El formato general de la instrucción es:

LOAD "CUADRADOS"

Proceso de grabación de un programa.



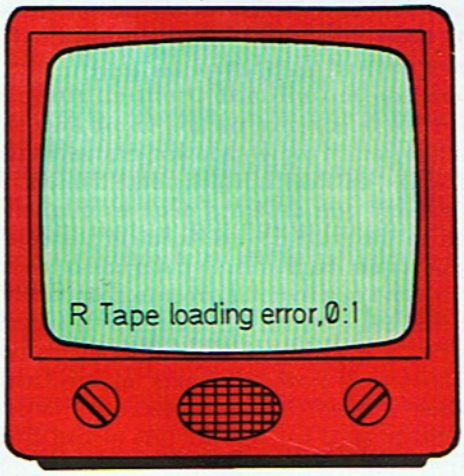
Lo primero que vamos a hacer para cerciorarnos de que el programa se carga desde la cinta, es borrar la copia que tenemos ahora en la memoria por medio de la instrucción **NEW**. Pulsamos después **NEW** y **ENTER**, para obtener el mensaje de presentación del ordenador. La palabra clave **NEW** está serigrafiada en color blanco sobre la tecla de la «A», por lo que podemos acceder a ella con la sola pulsación de dicha tecla, cuando nos encontramos en el modo **K**.



Hemos de tener mucho cuidado para no intentar grabar sobre la zona plástica, que inicia y termina las cintas de casete de uso habitual.

NEW, en inglés nuevo, indica al ordenador que vamos a introducir otro programa, por lo que queremos suprimir el que reside actualmente en la memoria, y preparar nuestro Spectrum para aceptar la carga desde casete, o introducción por el teclado, de un nuevo programa.

Basándonos en las franjas que preceden a la cabecera, podemos controlar el volumen adecuado del casete.



El mensaje R Tape loading error, 0:1 *aparece cuando se produce algún error durante la carga o verificación de un programa.*

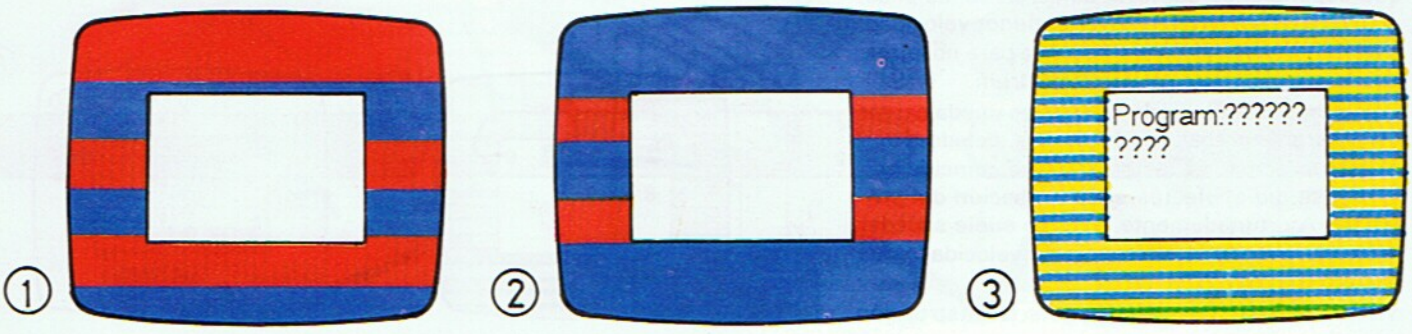
Para cargar el programa "CUADRADOS" desde el casete, lo primero que debemos hacer es situarnos en el lugar adecuado de la cinta. Podemos conseguirlo con la ayuda del cuentavueltas, o por medio de la orientación que nos proporciona, al rebobinar rápidamente la cinta con salida por el altavoz, nuestra voz grabada al comienzo de los programas. Una vez situados correctamente, podemos introducir la siguiente instrucción:

LOAD "CUADRADOS"

En la pantalla obtendremos exactamente los mismos resultados, que los comentados al hablar de la verificación (**VERIFY**). Primero el marco de la pantalla oscilando entre los tonos rojo y azul claro, y después las bandas anchas de estos colores, seguidas por el nombre del programa de la forma:

Program: CUADRADOS

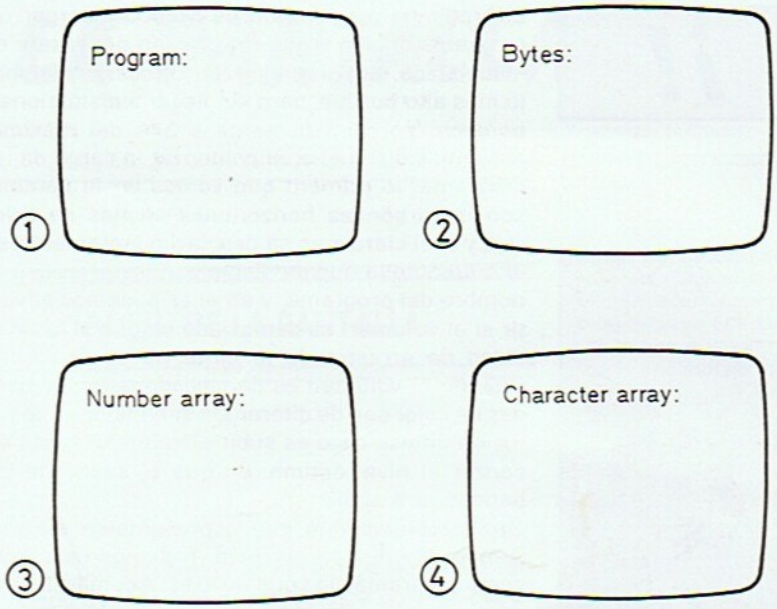
Seguirán nuevamente las franjas anchas de color, y por último las bandas finas azules y amarillas, indicativas de que el Spectrum está leyendo el bloque principal del programa. Cuando el ordenador concluye la carga, nos envía el men-



saje de **OK** y, pulsando **ENTER**, podemos comprobar que el programa está ya copiado en la memoria, de la misma forma que antes de ejecutar la sentencia **NEW**.

OMISION DEL NOMBRE DE PROGRAMA

Los nombres de los programas están formados por series de hasta 10 caracteres de longitud, compuestas por cualquier combinación de letras mayúsculas o minúsculas, números, espacios en blanco y caracteres especiales. Tanto para la sentencia **LOAD** como para **VERIFY** puede omitirse el nombre del programa, tecleando: **LOAD ""** o **VERIFY ""**, respectivamente.
En ambos casos, el Spectrum supone que queremos cargar o verificar el programa que encuentre grabado inmediatamente a continuación en la cinta. Esto quiere decir que, si no especificamos un nombre, el ordenador no va a comprobarlo, por lo que debemos estar seguros de que el programa que se encuentra a continuación es el que deseamos. De no ser así, cargaríamos en memoria otro programa, o verificaríamos uno distinto del contenido en la memoria, con el consiguiente error.
Por el contrario, cuando introducimos un nombre de programa acompañando a la palabra clave **LOAD**, el Spectrum comprueba, al poner en marcha la cinta, los nombres de los programas que va encontrando, cargando el que coincide con el especificado. Más concretamente, el primero que comience por la secuencia de caracteres que componen el nombre del programa. Si en la cinta hemos grabado varios programas con el mismo nombre, el Spectrum cargará el primero de ellos.

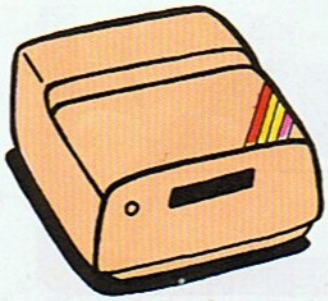


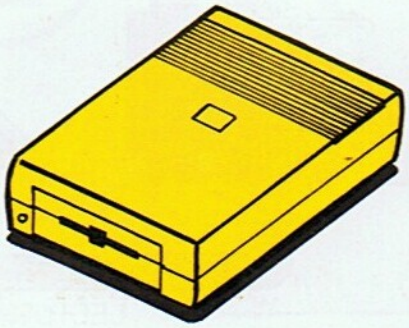
Son cuatro los mensajes que pueden aparecer en la pantalla tras la lectura de una CABECERA.

SOLUCION A POSIBLES PROBLEMAS

Existen dos causas principales por las que el ordenador no conseguirá efectuar la operación de carga correctamente. La primera de ellas se refiere a problemas de volumen del casete; la segunda, a que pretendemos que el ordenador cargue un programa que realmente no lo es.

Existen varios sistemas para el almacenamiento de programas. El casete, aunque lento, es bastante fiable y de bajo coste.





¡!

El volumen para las operaciones de carga y verificación de programas debe ser lo más alto posible, pero sin llegar a que el casete distorsione el sonido. Generalmente el volumen más adecuado es unos 3/4 del máximo.

*

Cuando al tratar de cargar un programa aparece en la pantalla algo distinto del mensaje **Program:**, debemos desistir de cargarlo, porque no se trata realmente de un programa sino de otra estructura de datos.

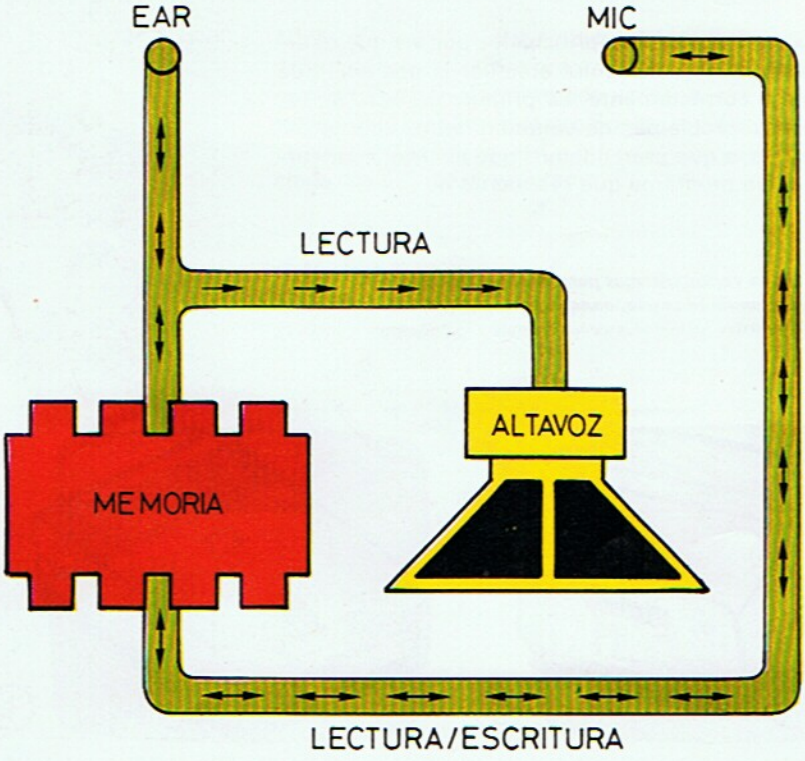
El problema más común es no poder cargar un programa debido a que el volumen del casete es inadecuado. El volumen debe colocarse siempre lo más alto posible, pero sin llegar a distorsionar, es decir, aproximadamente a 3/4 del máximo. Ya hemos dicho que, al comenzar la carga de un programa, lo primero que vemos en la pantalla son unas bandas horizontales anchas de color rojo y azul claro, que se desplazan lentamente en el marco de la misma. Estas bandas preceden al nombre del programa, y en ellas podemos advertir si el volumen es demasiado bajo, o si la velocidad de arrastre de la cinta no es uniforme. Cuando el volumen es demasiado bajo, las bandas de color son de diferentes dimensiones; la solución en ese caso es subir el volumen hasta alcanzar el nivel óptimo, en que el ancho de las bandas será similar.

Otro problema que puede presentarse es que, siendo el volumen correcto, tratemos de cargar como programa algo que no lo es. Además de programas, el Spectrum puede almacenar en la cinta, datos de código máquina, pantallas y datos numéricos o alfanuméricos.

Antes hemos visto que al cargar o verificar el programa "CUADRADOS", el Spectrum imprime el mensaje:

Program: CUADRADOS

Aunque es aconsejable realizar las cargas y verificaciones mediante el EAR *del Spectrum, y las grabaciones a través del* MIC, *cada una de las salidas puede cumplir la función de la otra.*



Además de la palabra **Program** (programa) que ya nos es familiar, puede ser que al intentar cargar una cinta comercial con **LOAD ""**, aparezca algún mensaje diferente. Eso quiere decir que la información que viene a continuación no se trata exactamente de un programa y, por lo tanto, debemos renunciar de momento e investigar sobre su contenido hasta más adelante. Los mensajes que pueden aparecer son:

Bytes:, **Character array:** o **Numeric array:**

UN EJERCICIO

Para llevar a cabo el ejercicio que proponemos, debemos borrar en primer lugar el programa que tenemos en la memoria; para ello, utilizaremos el comando **NEW**, que ya hemos estudiado anteriormente. Después vamos a introducir un nuevo programa, que puede ser cualquiera de los propuestos anteriormente; el "CUADRADOS", por ejemplo.

La primera sentencia con la que vamos a practicar es **SAVE**, que permitirá almacenar el programa en la cinta con el nombre que nosotros indiquemos. Para preparar la grabación del programa, debemos recordar colocar la cinta en un sitio donde podamos grabar y anotar el número de vuelta, así como registrar con nuestra voz el nombre que le demos al programa. Acto seguido, escribimos:

SAVE "nombre-programa"

El siguiente paso va a ser efectuar la verificación del programa. Para preparar la verificación, rebobinamos la cinta hasta el número de vuelta donde sabemos que comienza el mismo, y utilizamos la palabra clave **VERIFY**, seguida del nombre que hemos dado al programa entre comillas. Como ya sabemos, un método alternativo es escribir simplemente:

VERIFY ""

No tendremos problemas, ya que como se trata del programa que acabamos de grabar, al rebobinar la cinta nos posicionaremos justamente antes de su comienzo. Si todo ha marchado correctamente, se llevará a cabo la verificación, y el mensaje **OK** aparecerá en la última línea de la pantalla.

Como último paso, debemos efectuar un **NEW** para borrar el programa contenido en la memoria y, seguidamente, intentar cargarlo de nuevo con la palabra clave **LOAD**, seguida del nombre del programa (simbólicamente "nombre-programa"), o bien, mediante **LOAD ""**.

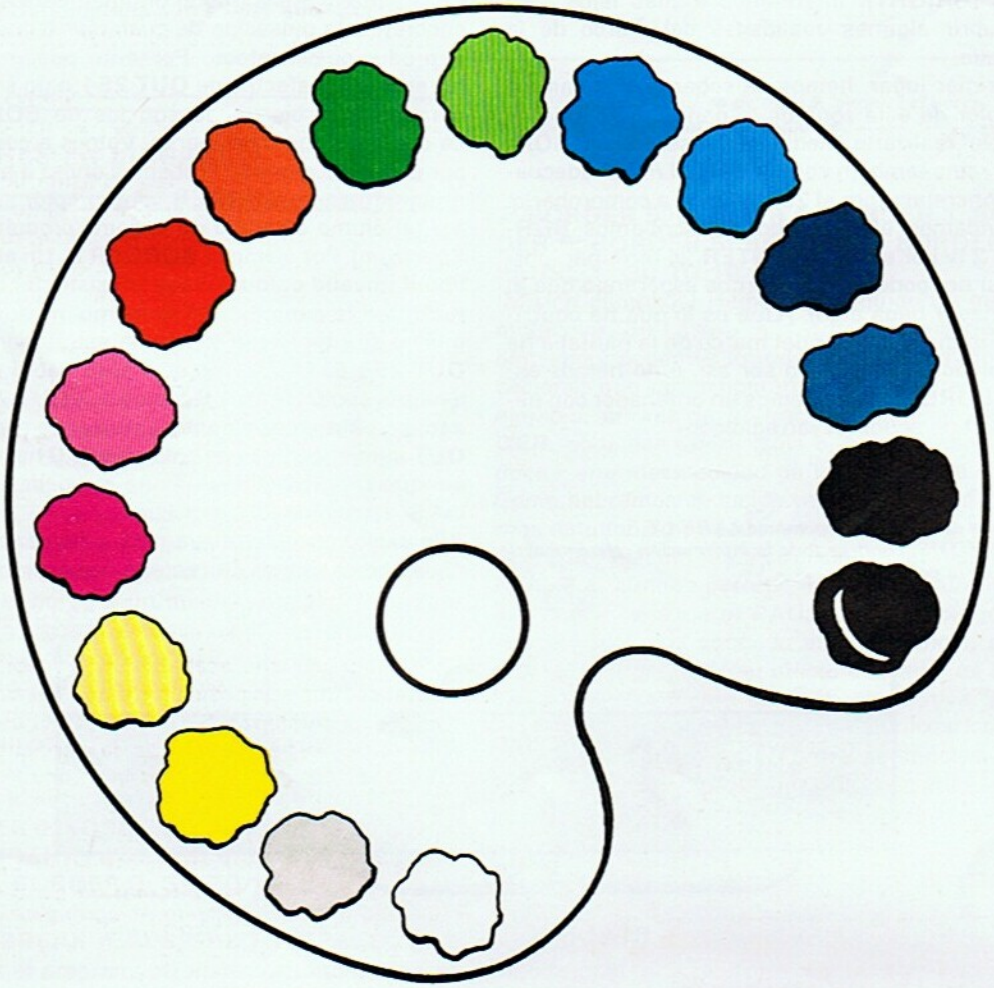
EL COLOR DE LA PANTALLA

EN nuestra conciencia está la necesidad de crear bonitas pantallas. Como ya vimos anteriormente, además de un diseño adecuado de las figuras que la integran, en esta tarea influyen además otros factores, tales como los colores empleados o el dinamismo de la pantalla en sí. A continuación daremos un pequeño repaso a varias técnicas basadas en el color de nuestro Spectrum, gracias a las cuales podremos conseguir interesantes efectos.

EL «BORDER» DE LA PANTALLA

Quizás podamos pensar hasta que hayamos terminado de leer este apartado, que poco se puede decir del marco de la pantalla. Nos equivocamos. Muchas sorpresas nos reserva nuestro aparato y entre ellas está la posibilidad de explotar algunas de sus sentecias BASIC, para obtener resul-

La gama de colores en el Spectrum es lo suficientemente variada en la mayoría de las ocasiones.



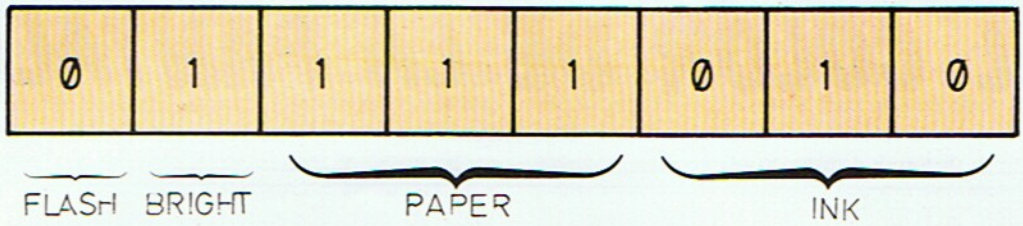
BITS

Mediante la combinación de colores en el interior de la pantalla y su marco, podemos obtener interesantes efectos ópticos. El siguiente programa es una clara demostración de este hecho.

```
10 FOR I=0 TO 7
20 PAPER 7-I: CLS: BORDER I
30 NEXT I: GO TO 10
```

*

El abecedario español incorpora una letra de relativamente frecuente utilización: la **Ñ**. Sin embargo, esta letra no se encuentra incluida en el juego de caracteres de nuestro Spectrum. Una forma muy sencilla de obtenerla, a partir del carácter gráfico de la **N**, es efectuar **POKE USR "n",BIN 001 11100** o bien **POKE USR "n",60.**



Cada byte del área de atributos, codifica la información sobre **PAPER, INK, BRIGHT** y **FLASH** *de cada carácter de la pantalla.*

tados que en un principio pueden parecer imposibles. Constataremos ahora este hecho a través de unos cuantos ejemplos.

Si aceptamos a «pies juntillas» lo que nuestro manual dice sobre el marco de la pantalla, y más concretamente sobre la sentencia **BORDER**, no cabe duda que tendremos la seguridad de no poder sacar gran partido de ese área de la pantalla. El «borde» será una zona monocolor en la que no nos es posible escribir texto o gráficos, ni tan siquiera otorgarle atributos propios de la pantalla principal, tales como intermitencia **(FLASH)**, o brillo **(BRIGHT)**. Intentemos ir más lejos para descubrir algunas cualidades del marco de la pantalla.

En primer lugar, hemos de saber que el cambio de color de esta zona de la pantalla, no sólo es posible realizarlo mediante la sentencia **BORDER**, sino también con un **OUT** al *PORT* adecuado, concretamente al 254. Vamos a comprobarlo: encendamos el ordenador y escribamos **BORDER 2** (y por supuesto **ENTER** al terminar, porque si no, podemos aburrirnos esperando que el ordenador haga algo). ¿Qué es lo que ha ocurrido? Sin duda el color del marco de la pantalla ha cambiado a rojo. De no ser así, o no hemos escrito **BORDER 2** o tenemos un ordenador con necesidad de acudir al «arreglólogo».

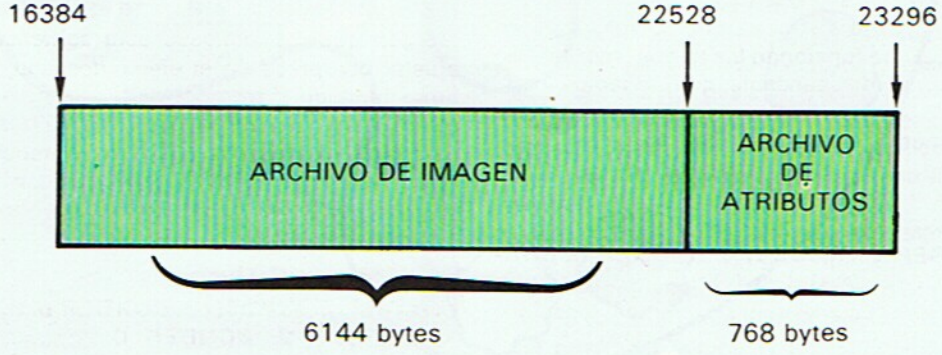
Ya podemos retornar el ordenador a su estado inicial; evidentemente utilizaremos una instrucción **BORDER 7.** A continuación, vamos a probar **OUT 254,2.** En apariencia todo ha sido igual; sin embargo, si somos un poco observadores, habremos percibido que mientras que el **OK** que sustituía a **BORDER 2** en la pantalla tenía el fondo rojo, el que ha sustituido a **OUT 254,2** no. Por tanto, la primera conclusión a la que llegamos, es que las líneas de edición toman el color del marco de la pantalla, que ha sido fijado mediante **BORDER**; sin embargo, la instrucción **OUT 254,n** no afecta a dichas líneas.

Pero aún existen más diferencias. Repite la operación anterior para retornar el color al blanco inicial. ¡Vaya sorpresa! ¿eh? No ha hecho falta más que escribir **BORDER** y el color del marco de la pantalla ha vuelto al original. Para ser más concretos, la pulsación de cualquier tecla hubiera producido este efecto. Por tanto, podemos asegurar que los efectos de **OUT 254,n** no son permanentes, si bien sí lo son los de **BORDER.**

La cosa se pone interesante. Vamos a continuar por el mismo camino. Probemos ahora a ejecutar una instrucción **BORDER** con un color superior a 7 (el último admitido en la gama cromática del Spectrum). Por ejemplo: **BORDER 8.** Un error del tipo **K (invalid colour** = color inexistente) ha aparecido en la pantalla. Sin embargo, no sucede lo mismo si intentamos la misma estratagema con **OUT 254,8.** El color en este caso habrá pasado a negro; con **OUT 254,9** a azul, con **OUT 254,10** a rojo, y así sucesivamente. Esto quiere decir, que **OUT** admite códigos de color desde 0 hasta 255, aunque simplemente se limita a repetir la gama del Spectrum de forma cíclica.

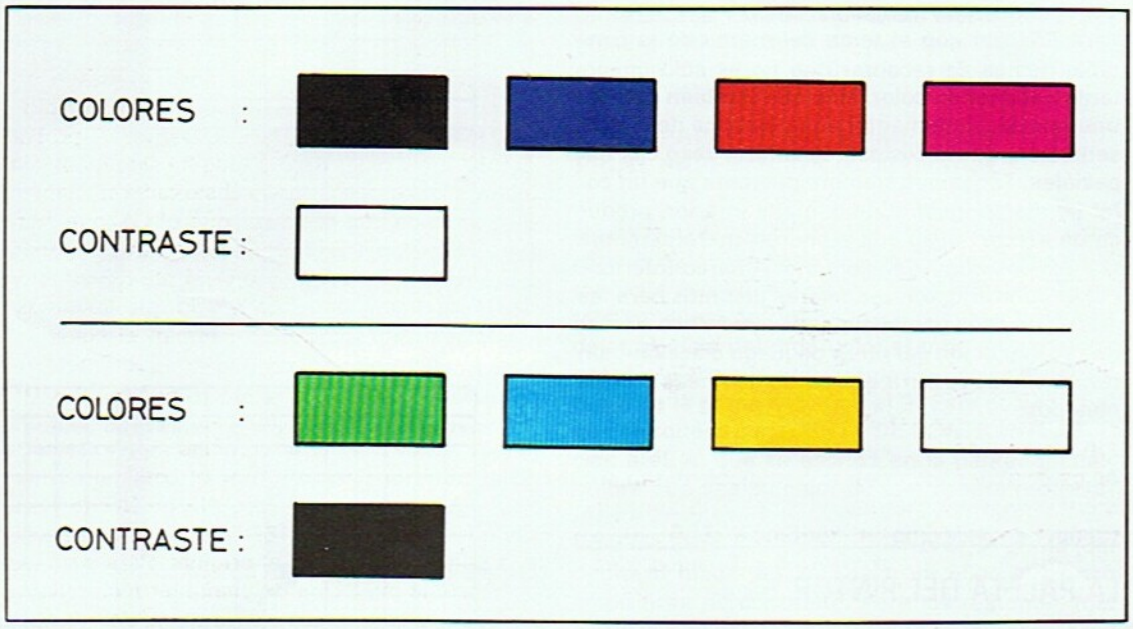
Aún existe otra diferencia más, esta vez relacionada con el sonido. Durante el transcurso de las

La zona de memoria denominada AREA DE ATRIBUTOS, contiene toda la información referente al color de la pantalla.



pruebas que hemos realizado, no hemos percibido ninguna vibración en el altavoz al ejecutar una sentencia **OUT 254, n.** Para ello será necesario que ejecutemos varias veces consecutivas el mismo trabajo, alternando los colores del marco: **10 FOR I=0 TO 255: OUT 254,I: NEXT I.**

nítidamente varios colores en el marco; sin embargo, el movimiento de las franjas de color, no nos han dejado disfrutar plenamente del espectáculo. Una temporización adecuada nos sacará de apuros. Incluyamos un retardo mínimo en la instrucción anterior: **10 BORDER 0: BORDER**



La utilización del código de tinta 9, permite el acceso a los colores de contraste del Spectrum.

COLORES EN EL MARCO

Vamos a abandonar ya el tema del **OUT**, pero no el del marco de la pantalla. El último experimento nos ha descubierto algo más que la emisión de sonido simultánea al color: hemos visto como el marco de la pantalla adquiría un tono indefinido y cambiante. Este es el primer paso hacia nuestro próximo objetivo: conseguir varios colores en el borde a un mismo tiempo. El sistema para conseguirlo, se basa en el «barrido» de pantalla con el que funcionan los televisores; en todo caso, no es indispensable que sepamos nada acerca de sus fundamentos y bastará con que aprendamos la técnica para conseguir este efecto.

Vamos a partir de la siguiente instrucción **10 BORDER 0: BORDER 2: BORDER 5: BORDER 6: BORDER 1: BORDER 6: BORDER 5: BORDER 2: BORDER 0: GO TO 10**

Cuando salgamos de nuestro asombro podemos pulsar **BREAK (CAPS SHIFT + SPACE)**, para detener el programa. Hemos apreciado bastante

2: BORDER 5: BORDER 6: BORDER 1: BORDER 6: BORDER 5: BORDER 2: BORDER 0: PAUSE 1: GO TO 10.

¡Misión cumplida! Hemos conseguido un marco multicolor. En base a la instrucción que hemos empleado, podemos introducir pequeñas modificaciones que varíen ligeramente el resultado. Por ejemplo, en vez de utilizar 7 sentencias **BORDER**, podemos utilizar sólo 5, aunque esto producirá una inestabilidad de las franjas de color. Como habremos deducido ya, el efecto que estamos estudiando se basa en el tiempo en que el Spectrum ejecuta las instrucciones; el resultado cambiará, aunque parezca mentira, según el lugar donde se sitúe el **PAUSE**. Sin embargo, el margen que nos concede esta sentencia, para ajustar con precisión el efecto deseado, es bastante limitado. Como solución alternativa, para retardos muy pequeños, podemos utilizar los separadores dos puntos (:) entre sentencias. Por ejemplo, vamos a conseguir gracias a ellos, deslizar suavemente por la pantalla las 7 franjas que generan nuestra instrucción base:

10 BORDER 0: BORDER 2: BORDER 5: BORDER 6: BORDER 1: BORDER 6: BORDER 5: BORDER 2: BORDER 0::::::::::::::::::::::::: GO TO 10

BITS

Una forma habitual de enfatizar determinadas palabras, es subrayarlas. A continuación listamos una interesante rutina, que facilita el subrayado de mensajes en la pantalla del Spectrum.

```
10 DEF FN Y$(X$)=X$(N)+CHR$ 8+CHR$ 21+CHR$ 1+"_"
20 INPUT X$: FOR N=1 TO LEN X$
30 IF X$(N)=" " THEN PRINT " ":: GO TO 50
40 PRINT FN Y$(X$);
50 NEXT N
```

Son muchas las sorpresas que nos reserva la ROM de nuestro Spectrum. Por ejemplo, seguro que muchos de nosotros desconocíamos que la acción simultánea sobre las teclas **CAPS SHIFT, SYMBOL SHIFT** y **Z**, fuerza la detención del programa, de la misma manera que lo hace **BREAK (CAPS SHIFT+SPACE)**.

BITS

Acudiendo a la ROM podemos solucionar muchos problemas que implican el manejo del código máquina. Tal es el caso de la subrutina de **scroll**, que se encuentra localizada en la dirección decimal 3190. Efectuando **RANDOMIZE USR 3190**, conseguiremos realizar un desplazamiento de la pantalla. Para comprobarlo introduzcamos el siguiente programa.

```
10 FOR I=0 TO 25
20 PRINT AT 21,0;"LINEA No.";I
30 PAUSE 30: RANDOMIZE USR 3190: NEXT I
```

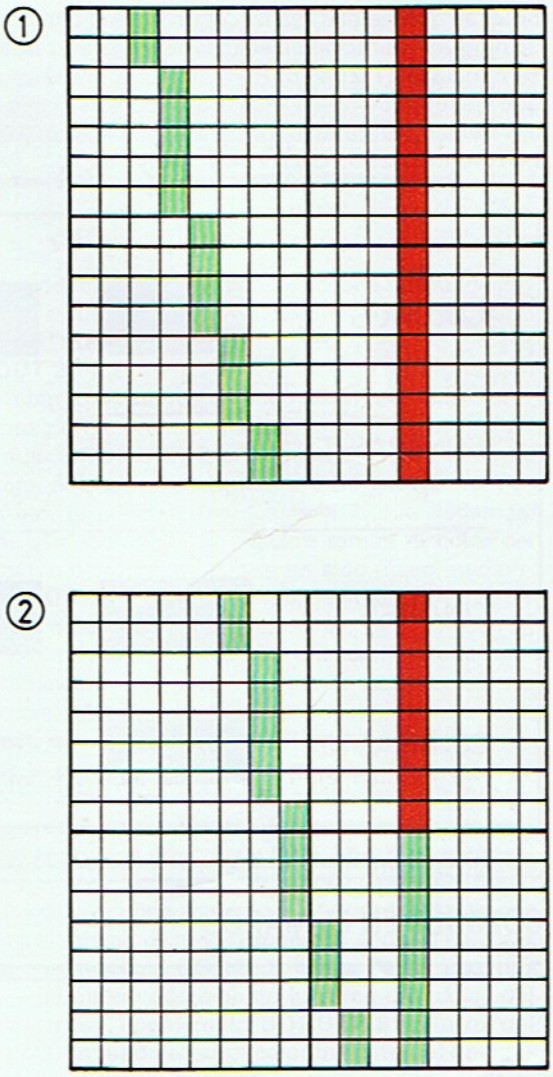
Para ello, hemos introducido 27 separadores entre **BORDER 0** y **GO TO 10.** Si nos dedicamos a hacer pruebas, comprobaremos que hasta la inclusión de un separador más o menos, influye en el resultado final. Utilizando código máquina, se dispone de una temporización extraordinariamente precisa, gracias a lo cual se pueden conseguir gran cantidad de franjas estables.

Para finalizar con el tema del marco de la pantalla, hemos de recordar que no es sólo importante cambiar su color, sino que también cumple una misión fundamental en la estética de la presentación, sin necesidad de utilizar técnicas especiales. Tengamos siempre presente que un color de marco igual al de pantalla interior, produce un efecto óptico que aumenta aparentemente las dimensiones del conjunto «marco-interior». Por el contrario, utilizar colores distintos para las dos zonas de la pantalla, puede ayudarnos a crear una delimitación del «área de juego o acción», sin necesidad de recurrir al trazado de líneas u otros métodos.

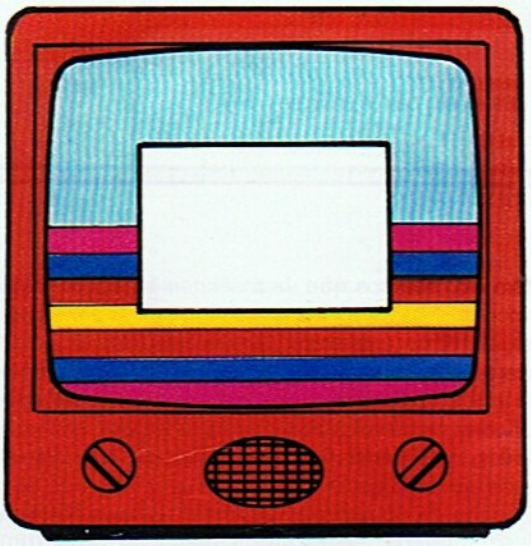
LA PALETA DEL PINTOR

Todos los pintores tienen su paleta de colores y nosotros, los diseñadores de pantallas, no vamos a ser menos. Para nuestra desgracia, los artistas del pincel electrónico no disponemos en el Spectrum de una gran variedad de tonalidades; sin embargo, esto no debe ser disculpa para no combinar adecuadamente aquellos colores que sí podemos emplear.

Para empezar, nunca debemos despreciar las grandes posibilidades que aporta el modo **BRIGHT**, ya que, en muchas ocasiones, las diferencias entre los tonos con y sin brillo son ostensibles. Veamos un ejemplo: encendamos el ordenador y ejecutemos la instrucción **BRIGHT 1: CLS.** Aparentemente, la pantalla ha continuado siendo blanca y el marco ha adquirido un color gris claro. Este efecto óptico se debe a la diferencia de brillo entre ambas zonas, y por supuesto lo podemos explotar a nuestra conveniencia. Lo mismo sucede cuando combinamos ambos brillos de amarillo, gracias a lo cual el ama-



La representación de dos líneas con distinto color de primer término, sólo es posible cuando están controladas por diferentes bytes del área de atributos, es decir, cuando pertenecen a dos caracteres diferentes.



Es posible, de una manera muy sencilla, obtener marcos de pantalla multicolor.

rillo que normalmente utilizamos adquiere un tono verdoso.
Efectos similares se pueden obtener con todos los colores, aunque según vamos adentrándonos en la gama, menor es la diferencia entre ambas tonalidades de cada color, hasta llegar al negro, donde la diferencia se hace prácticamente imperceptible.
Una vez conscientes de que nuestra paleta no se reduce a 8 tonos, sino a 16. Hemos de manejar con cuidado los colores de fondo y primer término, es decir, **PAPER** e **INK**, respectivamente. Nunca debemos utilizar colores de una excesiva proximidad en la gama, pues a no ser que estemos tratando grandes superficies, la visibilidad del primer término sobre el fondo se hace bastante escasa. Este desagradable efecto, queda patente en la escritura de mensajes. He aquí un ejemplo clarísimo de lo que estamos diciendo:

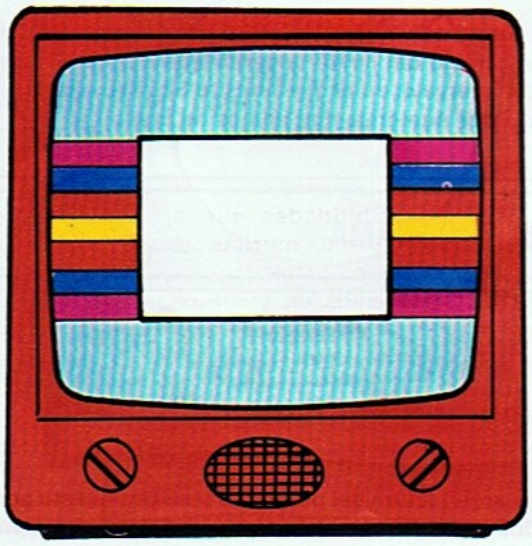
```
PRINT PAPER 7; INK 6;"MENSAJE"
PRINT PAPER 7; INK 0;"MENSAJE"
```

No debe existir duda alguna entre cual de los dos colores ha conseguido una mejor visibilidad en el mensaje escrito. Para evitar este tipo de «desastres visuales» de una manera cómoda, podemos dejar a juicio del ordenador el tono en que debe escribir. Para ello utilizaremos un color especial, fuera de la gama habitual, el **9**. Este código es el que se conoce como TINTA DE CONTRASTE. Gracias a él, lo que se escriba en la pantalla, cualquiera que sea su color de fondo, será perfectamente legible.
Para los cuatro primeros colores de la gama (del negro al magenta), el Spectrum elige el blanco como tinta de contraste. Para los restantes (del verde al blanco), la tinta elegida es la negra. Este es el mismo sistema que el Spectrum sigue para escribir en la línea de edición. Con la siguiente rutina veremos como el ordenador ajusta el color de la tinta mediante el código 9.

```
10 FOR I=0 TO 7
20 BORDER I: PAPER I: CLS
30 PRINT INK 9;"MENSAJE"
40 PAUSE 100
50 NEXT I
```

De la misma manera que **INK 9** ajusta el color de primer término, como contraste con el de fondo, **PAPER 9**, contrasta éste con el de tinta. Pese a todo lo dicho, en ocasiones no es tarea fácil compatibilizar varios colores en puntos *(pixels)* muy cercanos, como sería nuestro deseo. Esto sucede frecuentemente al trazar líneas de diferente color con la sentencia **DRAW**.
Hay que tener en cuenta que a cada posición de carácter de la pantalla, le corresponde un solo byte de la zona de atributos y, por tanto, únicamente se puede combinar un color de fondo y uno de primer término en un mismo carácter. De igual modo, cualidades como el brillo afectan simultáneamente tanto a papel como a tinta. Así pues, la única solución para escribir dos puntos consecutivos con distinto tono de primer término, es situarlos en caracteres diferentes, con el fin de que sean controlados con bytes distintos del área de atributos.

Una pequeña diferencia en el tiempo de ejecución de las instrucciones generadoras del marco multicolor, produce un desplazamiento de las franjas de color.



BITS

He aquí un programa con el que podremos dejar admirados a nuestros amigos. Gracias a él, generaremos espectaculares gráficos de alta resolución, ¡empleando sólo tres instrucciones!

```
10 PLOT 90,80
20 INPUT N
30 DRAW 60,60,N:
PAUSE 100: RUN
```

Dependiendo del valor que asignemos a la variable **N**, mediante el **INPUT** de la línea 20, obtendremos diferentes resultados. Podemos empezar probando con los siguientes valores: 7689, 8462, 3572, 1010, 5387, 3546, 2547, 4189 y 8888.

*

El efecto de parpadeo que produce la sentencia FLASH del Spectrum, carece en ocasiones de la estética deseable para la impresión de mensajes. Un metodo alternativo al FLASH del sistema, puede ser escribir y borrar el mensaje cíclicamente. Este efecto lo podíamos conseguir de una menera muy simple, utilizando repetidamente la sentencia **OVER**.

```
10 PRINT AT
11,6;OVER 1;"MEN-
SAJE PARPADEAN-
TE": PAUSE 25: GO
TO 10
```

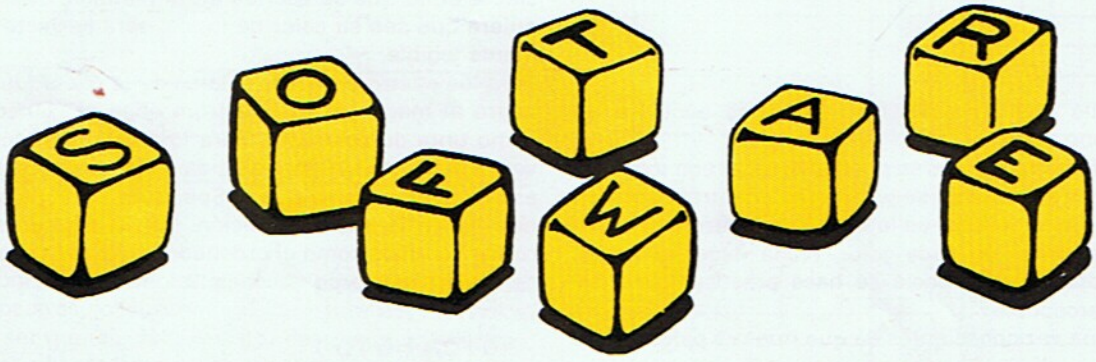
ADIVINADOR

EL programa que a continuación presentamos, se encuentra dentro del grupo de los llamados juegos de lógica; el problema que plantea es el siguiente: el ordenador nos reta a que adivinemos un número que él mismo ha «pensado». Dicho número está comprendido entre 1 y un valor determinado (número tope), prefijado por nosotros mismos. A partir de ese momento, comienza el juego, debiendo hacer un esfuerzo por adivinar la cifra propuesta en el menor número de intentos posible.

Pero no creamos que va a ser tan fácil. Por cada cinco intentos consecutivos sin conseguir desvelar el número secreto, tendremos que superar una prueba propuesta por el retorcido «cerebro» de nuestro Spectrum: debemos reconstruir una palabra previamente descompuesta por el ordenador, y disponemos para ello de sólo cinco oportunidades.

En caso de no ordenar correctamente la palabra desmantelada, en el límite fijado, el Spectrum elegirá otro nuevo «número incógnita» dentro del intervalo inicial predefinido, resultando por tanto baldíos nuestros anteriores intentos de adivinar el número. Por el contrario, si logramos dar con la palabra desordenada dentro de las cinco oportunidades que el ordenador nos concede, podremos continuar el juego donde lo habíamos dejado.

Una vez adivinado el número, podremos optar por el cambio del límite superior que define el tope del intervalo, en el que nuestro Spectrum escoge el valor numérico que hemos de descubrir.



Por cada cinco «ensayos adivinatorios» errados, el programa propone la recomposición de un puzzle de letras, con el tema informático de fondo.

Al empezar, el ordenador nos mostrará en la pantalla los valores que definen el intervalo en el que se encuentra el «número incógnita». Sólo nos resta adivinar el número elegido por el Spectrum. Para ello, iremos tanteando el intervalo en sucesivos intentos.

Por cada número introducido, aparecerá un mensaje que dependerá de la comparación lógica entre el valor a adivinar y el dato numérico anteriormente tecleado. Así pues, sólo pueden darse tres casos, y por tanto tres mensajes diferentes:

1. Número introducido = Número incógnita
   MENSAJE: **ACERTASTE**
2. Número introducido > Número incógnita
   MENSAJE: **INTRODUCE UN NUMERO MENOR**
3. Número introducido < Número incógnita
   MENSAJE: **INTRODUCE UN NUMERO MAYOR**

EL PROGRAMA

La arquitectura del programa está compuesta por una serie de subrutinas, que son llamadas bien

¡!

Para grabar el programa, utilizaremos la instrucción SAVE "ADIVINADOR" LINE 10. Si suprimiésemos la sentencia LINE 10, el programa no se autoejecutaría al finalizar la carga del mismo.

*

El vocabulario especificado en las dos primeras DATAs del listado, se refiere por entero a vocablos informáticos. Podemos variar el contenido de dichas DATAs, teniendo en cuenta que el número de palabras que hay en cada una de ellas ha de ser constante.

*

Las subrutinas de programa que son accedidas desde otras subrutinas, se denominan **subrutinas anidadas.**

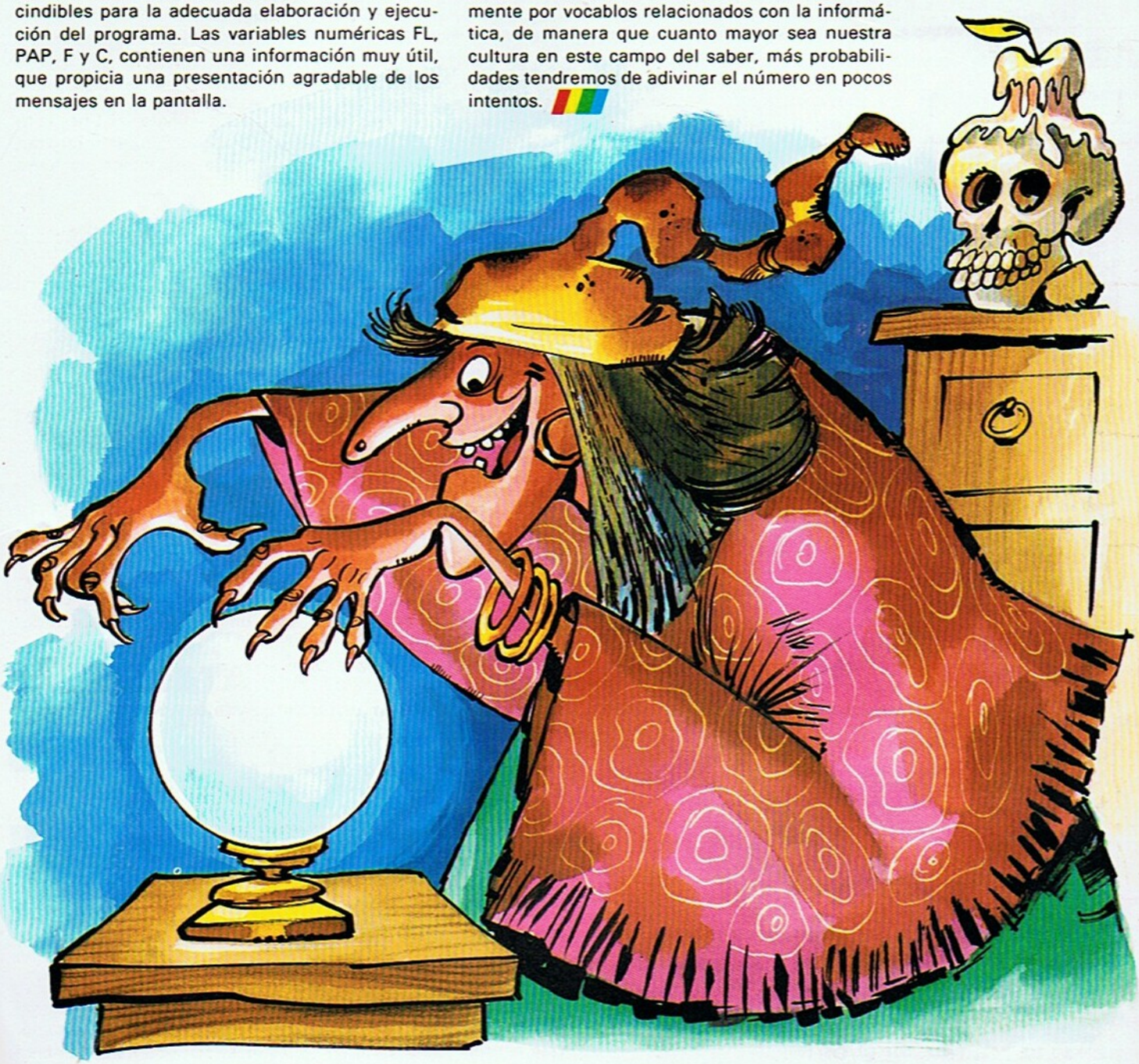
*

El puzzle de letras se desordena en la zona de programa 380-430, apoyándose en las cadenas A$, P$ y O$.

desde el programa principal, o bien desde otra subrutina **(subrutinas anidadas).**

Por lo que respecta a las variables, cabe observar que se reducen estrictamente a las imprescindibles para la adecuada elaboración y ejecución del programa. Las variables numéricas FL, PAP, F y C, contienen una información muy útil, que propicia una presentación agradable de los mensajes en la pantalla.

Finalmente, hay que destacar que el banco de datos, con el que el ordenador trabaja para proponer y desordenar palabras, reside en las dos primeras líneas **DATA**, y está compuesto enteramente por vocablos relacionados con la informática, de manera que cuanto mayor sea nuestra cultura en este campo del saber, más probabilidades tendremos de adivinar el número en pocos intentos.



```
 10 REM ***********************
 20 REM ***** J.M.MAYORAL *****
 30 REM ***********************
 40 PAPER 0: BORDER 0: CLS
 50 POKE 23658,8
 60 LET MU=1
 70 GO TO 890
 80 REM SBR.ELIGE NUMERO
 90 INPUT "NUMERO TOPE =  ";NT: GO SUB 1630
100 LET NUM=INT (RND*NT)+1
110 RETURN
120 REM
130 REM SBR.ESCRIT.HORIZ.
140 FOR N=1 TO LEN A$
150 PRINT  PAPER PAP; FLASH FL;AT F,C+N;A$(N)
160 NEXT N
170 RETURN
180 REM SBR.BORRADO
190 FOR N=1 TO 30
200 PRINT  PAPER 0;AT F,N;" "
210 NEXT N
220 RETURN
230 REM BORRADO PANTALLA
240 FOR F=1 TO 20
250 PRINT  PAPER 0;AT F,1;"
                                  "
260 NEXT F
270 RETURN
280 REM SBR.MULTIPLO DE 5
290 IF CONT=MU*5 THEN  LET MU=MU+1: GO TO 450
300 RETURN
310 DATA "ACUMULADOR","BYTE","OCTETO","BIT","COMPILA
DOR","FIRMWARE","BASIC","SOFTWARE","HARDWARE","CHIP",
"SENTENCIA","SCROLLING","BAUDIO","BUG","RAM","ROM","C
ODIGO","INTERFACE","BUCLE","HIT","CURSOR","CPU","INFO
RMATICA","INICIALIZACION","RUTINA","SUBPROGRAMA"
320 DATA "IMPRESORA","JOYSTICK","EDITAR","ITERACION"
,"MONITOR","MODEM","TERMINAL","VARIABLE","CAMPO","REG
ISTRO","MENU","KBYTE","PROGRAMA","MATRIZ","TABLA","EN
SAMBLADOR","DESENSAMBLADOR","SPRITE","WINCHESTER","HE
URISTICO","DOWNLOAD","ALGORITMO","PSEUDOCODIGO","ASCI
I","BOOTING","BOOTSTRAPPING","CARACTER","BUS","PORT",
"CANAL","PILA","PERIFERICO","PIXEL","PROMPT"
330 RESTORE 310
340 FOR N=1 TO INT (RND*60)+1
350 READ P$
360 NEXT N
370 DIM O$(LEN P$)
380 FOR N=1 TO LEN P$
390 LET L=INT (RND*LEN P$)+1
400 IF O$(L)<>" " THEN  GO TO 390
410 BEEP .1,20: LET O$(L)=P$(N)
420 IF O$=P$ THEN  BEEP 1,20: GO TO 310
430 NEXT N
440 RETURN
450 REM SBR.PASAT
460 GO SUB 230
470 LET A$="5 INTENTOS SEGUIDOS FALLADOS"
480 LET FL=1: LET PAP=2: LET F=10: LET C=1
490 GO SUB 140
500 LET A$="RECOMPON LA PALABRA..."
510 LET FL=0: LET PAP=2: LET C=4: LET F=12
520 GO SUB 140
530 GO SUB 310
540 LET A$=O$
550 LET FL=0: LET F=14: LET PAP=0: LET C=INT ((32-LE
N P$)/2)-2
560 GO SUB 140
570 PRINT AT 19,6;" OPORTUNIDADES = ";5
580 FOR X=4 TO 0 STEP -1
590 INPUT "LA PALABRA ES.. "; LINE I$
600 BEEP .1,30
610 IF I$=P$ THEN  GO TO 810
620 LET A$="PALABRA INCORRECTA"
630 LET PAP=2: LET FL=1: LET F=17: LET C=5
640 GO SUB 140: BEEP 1,-10: GO SUB 180
650 PRINT AT 19,6;" OPORTUNIDADES = ";X
660 NEXT X
670 PRINT  PAPER 0;AT 19,3;"
                                  "
680 LET A$="      LA PALABRA ES        "
690 LET PAP=6: LET F=17: LET C=2: LET FL=0
700 GO SUB 140
710 LET A$=P$
720 LET PAP=0: LET F=19: LET C=INT ((32-LEN P$)/2)-2
730 GO SUB 140
740 FOR G=1 TO 10: BEEP .1,40: BEEP .1,35: BEEP .1,3
0: NEXT G
750 PAUSE 100
760 GO SUB 230
770 LET A$="ELIJO OTRO NUMERO"
780 LET F=10: LET C=6: LET FL=0
790 GO SUB 140: PAUSE 50: GO SUB 180
800 LET CONT=0: LET MU=1: LET V=0: GO SUB 100: GO TO
910
810 LET V=1
820 GO SUB 230
830 LET A$="  CORRECTO  "
840 LET PAP=6: LET FL=1: LET F=10: LET C=9
850 GO SUB 140
860 BEEP .5,10: BEEP .5,20
870 GO SUB 180
880 RETURN
890 REM PROGRAMA PRINCIPAL
900 GO SUB 80
910 PRINT  INK 6; PAPER 5;AT 21,0;"
                                  "
920 PRINT  INK 6; PAPER 5;AT 0,0;"
                                  "
930 INK 9
940 FOR F=1 TO 20
950 PRINT  PAPER 5; INK 6;AT F,0;" ";AT F,31;" "
960 NEXT F
970 PRINT AT 7,6;"EL NUMERO QUE DEBES"
980 PRINT AT 9,8;"ADIVINARME ESTA"
990 PRINT AT 11,6;"COMPRENDIDO ENTRE..."
1000 PRINT  PAPER 2;AT 13,11;"1"; PAPER 0;"  Y  "; PA
PER 2;NT
1010 GO SUB 1560
1020 FOR N=1 TO 250
1030 NEXT N
1040 LET A$="PULSA UNA TECLA PARA EMPEZAR"
1050 LET PAP=0: LET FL=0: LET F=19: LET C=1
1060 GO SUB 140
1070 IF INKEY$="" THEN  GO TO 1070
1080 BEEP .3,50
1090 GO SUB 230
1100 LET CONT=0
1110 LET A$="INTRODUCE NUMERO"
1120 LET PAP=0: LET FL=1: LET F=19: LET C=7
1130 GO SUB 140
1140 INPUT NC
1150 PRINT  PAPER 0;AT 5,1;"
                                  "
1160 LET K$=STR$ NC
1170 LET F=19: GO SUB 180
1180 LET F=10: GO SUB 180
1190 LET CONT=CONT+1
1200 IF NUM>NC THEN  GO TO 1230
1210 IF NUM<NC THEN  GO TO 1300
1220 IF NUM=NC THEN  GO TO 1370
1230 REM NUM>NC
1240 GO SUB 280
1250 LET A$="INTRODUCE UN NUMERO MAYOR"
1260 LET PAP=6: LET FL=0: LET F=10: LET C=2
1270 GO SUB 140
1280 PRINT AT 5,INT ((32-LEN K$)/2);NC
1290 GO TO 1110
1300 REM NUM<NC
1310 GO SUB 280
1320 LET A$="INTRODUCE UN NUMERO MENOR"
1330 LET PAP=2: LET FL=0: LET F=10: LET C=2
1340 GO SUB 140
1350 PRINT AT 5,INT ((32-LEN K$)/2);NC
1360 GO TO 1110
1370 REM NUM=NC
1380 GO SUB 230
1390 LET A$=" A C E R T A S T E "
1400 LET PAP=5: LET FL=1: LET F=10: LET C=5
1410 GO SUB 140
1420 LET A$="LO ADIVINASTE EN "
1430 LET PAP=0: LET FL=0: LET F=19: LET C=1
1440 GO SUB 140
1450 PRINT  PAPER 6;AT 19,19;CONT; PAPER 0;" INTENTOS
"
1460 FOR N=-20 TO 40
1470 BEEP .01,N
1480 BORDER 5: BORDER 2: BORDER 1: BORDER 6
1490 NEXT N
1500 BORDER 0
1510 GO SUB 230
1520 PRINT AT 19,1;"DESEAS CAMBIAR EL NUMERO TOPE?"
1530 IF INKEY$="S" THEN  RUN
1540 IF INKEY$="N" THEN  GO SUB 230: GO SUB 100: LET
MU=1: LET CONT=0: GO TO 910
1550 GO TO 1530
1560 RESTORE 1570
1570 DATA 74,46,32,77,46,32,77,65,89,79,82,65,76
1580 FOR N=1 TO 13
1590 READ A
1600 PRINT  INK 5;AT 2,N+8;CHR$ A
1610 NEXT N
1620 RETURN
1630 FOR N=1 TO 10
1640 LET L=RND*ABS (SQR N*NT)
1650 NEXT N
1660 RETURN
```

UN POCO DE PRACTICA

VAMOS a aplicar los conocimientos adquiridos, para confeccionar un programa que nos permita calcular la nota media obtenida en una determinada asignatura, suministrando al ordenador el número total de calificaciones y cada una de estas en particular.
Comenzamos preparando el Spectrum por medio de la instrucción **NEW** para aceptar este nuevo programa y tecleamos:

```
 10 REM - CALCULO DE NOTA MEDIA -
J.M.LOPEZ MARTINEZ
 20 LET T=0
 30 LET I=0
 40 CLS
 50 INPUT "Numero total:";N
 60 PRINT "Numero total: ";N
 70 PRINT
 80 PRINT "--- CALIFICACIONES OBTENIDAS ---"
 90 INPUT C
100 PRINT C,
110 LET T=T+C
120 LET I=I+1
130 IF I<N THEN  GO TO 90
140 PRINT
150 LET M=T/N
160 PRINT "Nota media..: ";M
170 INPUT "Fin de programa ?";F$
180 IF F$="NO" THEN  GO TO 20
190 IF F$<>"SI" THEN  GO TO 170
```

REM

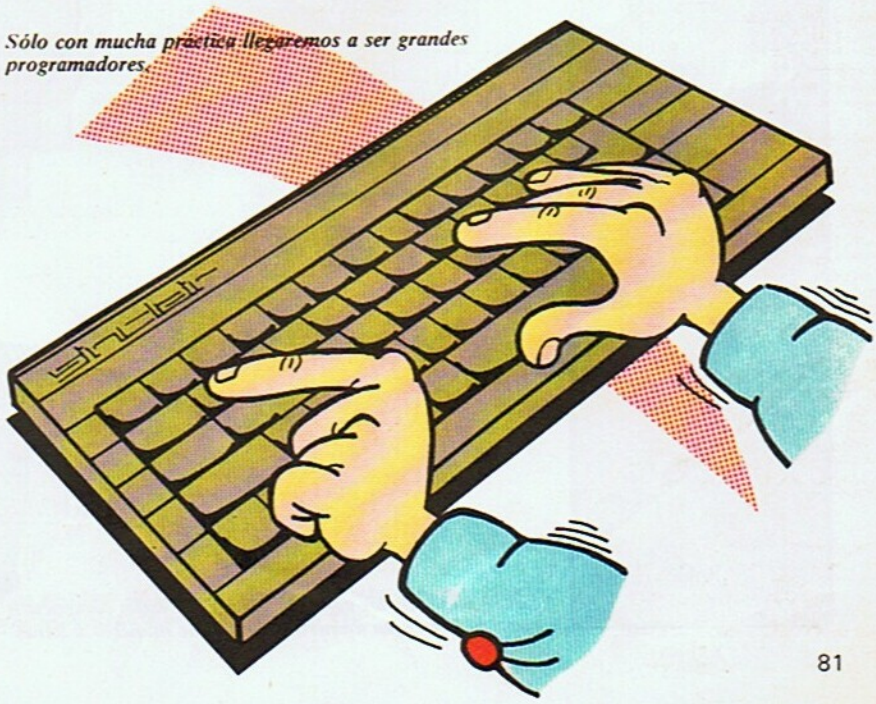
La palabra clave BASIC **REM** no nos ha sido todavía presentada. Para remediar este olvido diremos que en inglés significa comentario **(REMARK)** y no es, por lo tanto, una instrucción para que el ordenador la ejecute, sino que nos sirve de ayuda para incluir comentarios clarificadores de lo que el programa hace, en los lugares que nos interese, o simplemente para indicar la utilidad del programa al principio de éste, como es el caso del ejemplo que nos ocupa:

10 REM - CALCULO DE NOTA MEDIA - J.M. LOPEZ MARTINEZ

Todo lo que se encuentra a la derecha de la palabra **REM** no es interpretado como una instrucción por el ordenador. Debido a esta particularidad, es una técnica de uso común entre los programadores, anteponer la plabra clave **REM** a una instrucción que queremos que el ordenador no considere, sin suprimirla definitivamente del programa:

20 REM LET T=0

Cuando estamos realizando pruebas con un determinado programa, y pensamos que sería necesario eliminar una instrucción, pero no estamos completamente seguros de que con ello vayamos a solucionar nuestros problemas, antes de suprimir la instrucción debemos anteponerle un **REM.**
Hecho esto, volvemos a ejecutar el programa y comprobamos sus resultados en la seguridad de que, si ahora las cosas marchan correctamente, nos basta con borrar definitivamente la instrucción tecleando su número seguido de **ENTER,** como ya vimos anteriormente.
Si, por el contrario, la instrucción debe permanecer aún en el programa, no tendremos que volver a teclearla, como en el caso de que la hubié-



Sólo con mucha práctica llegaremos a ser grandes programadores.

¡!

La palabra BASIC **REM** sirve para introducir comentarios en los programas, y no es ejecutada como una instrucción por el ordenador.

*

El efecto de cualquier instrucción precedida de **REM** queda absolutamente anulado, por lo que suele ser empleada en la depuración de programas.

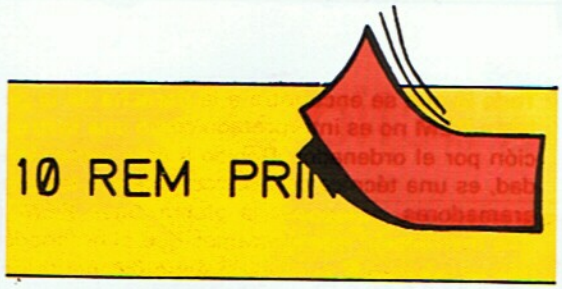
*

En una instrucción de asignación **LET**, siempre se calcula el valor de la expresión a la derecha del símbolo igual (=), y se asigna a la variable situada a la izquierda del mismo.

*

Con las variables numéricas se pueden realizar cualquier tipo de operaciones matemáticas, pero no con las alfanuméricas.

ramos suprimido directamente, sino que nos bastará con desproveerla de su **REM** para que el programa vuelva a quedar en las mismas condiciones que al principio. Digamos, por tanto, que **REM** tiene el efecto de «anular» las instrucciones a las que se antepone.



El REM se utiliza para anular el efecto de las instrucciones a que se antepone.

LET

La instrucción **LET** asigna valores a las variables BASIC. La estructura de esta sentencia es siempre la misma. La instrucción comienza por un **LET**, que va inmediatamente seguido por el nombre de la variable a que afecta y el signo igual (=). A la derecha del símbolo de igualdad se encuentra el nombre de otra variable, una constante o una expresión que el ordenador va a calcular para determinar el valor de la variable especificada por **LET**:

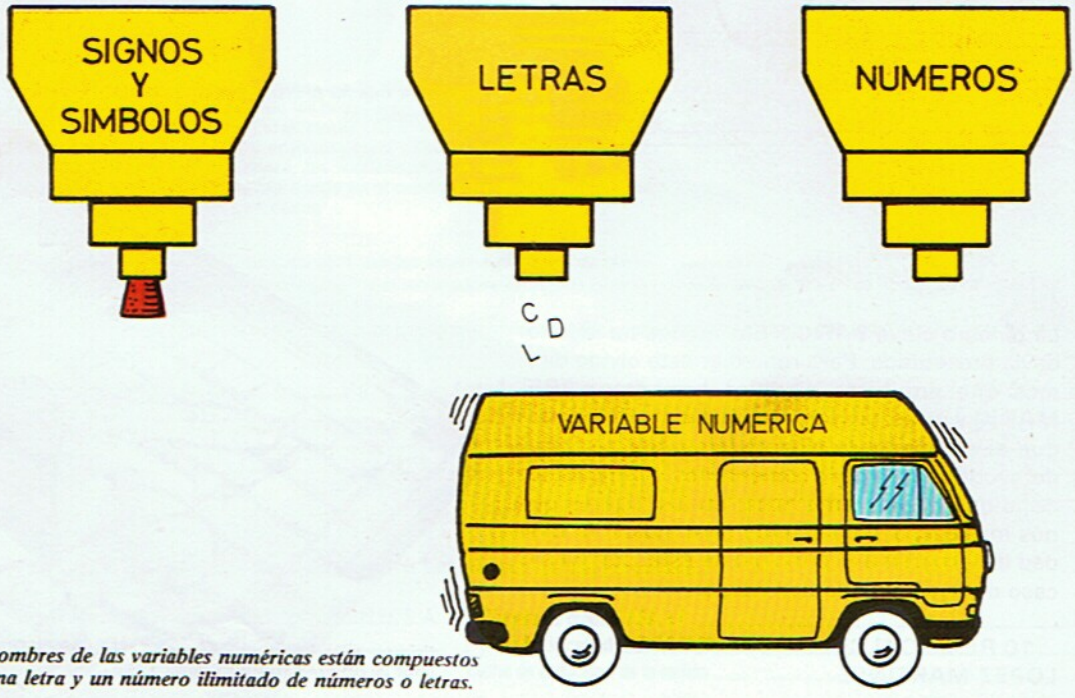
```
20 LET T=0
30 LET I=0

110 LET T=T+C
120 LET I=I+1
150 LET M=T/N
```

En el caso de las líneas 20 y 30, igualamos las variables especificadas por **LET** a otras variables (del mismo tipo), situadas a la derecha del signo igual (=). Este es el sistema más simple de asignación.

En las instrucciones 110, 120 y 150, la asignación se efectúa por medio de una fórmula. El ordenador debe calcular, en estos casos, el resultado de la expresión a la derecha del símbolo igual (=), para dar después el valor obtenido a la variable correspondiente.

A la hora de asignar valores a las variables, debemos poner especial atención en el tipo de variables a que nos estamos refiriendo. Ya hemos dicho que el Spectrum es capaz de manejar tanto números como letras; sin embargo, el ordenador no da el mismo tratamiento a los caracteres numéricos que a los alfabéticos. Esta distinción es lógica porque, si pensamos un poco, nos damos cuenta en seguida de que con los números podemos realizar cálculos, pero no con las letras. Con más propiedad, deberíamos decir que los dos tipos de variables que existen son las «numéricas» y las «alfanuméricas».

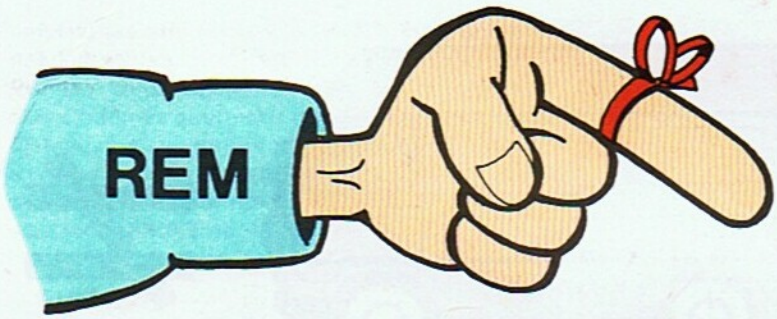


Los nombres de las variables numéricas están compuestos por una letra y un número ilimitado de números o letras.

Las variables numéricas deben contener forzosamente números y se emplean para realizar cálculos de tipo numérico. Se denominan con una serie de caracteres alfanuméricos, el primero de ellos alfabético. Con ellas se pueden realizar, por lo tanto, las operaciones matemáticas básicas, o de cualquier tipo más complejo.
Las variables alfanuméricas pueden contener números, letras o caracteres especiales, es decir, cualquiera de los accesibles por medio del teclado, incluyendo símbolos gráficos. Estas variables se denominan con una letra más el símbolo dólar **($)**, y no pueden ser sometidas a las operaciones matemáticas, pero si admiten la CONCATENACION y el TRUNCAMIENTO.
La concatenación consiste en la «suma» de dos o más cadenas de caracteres para dar lugar a una nueva, mientras que el truncamiento nos permite obtener una subcadena a partir de una cadena de caracteres dada; es decir, utilizar sólo una parte de la cadena original. Estas dos operaciones con cadenas serán estudiadas con detalle más adelante.
Debemos tomar buena nota de que no podemos mezclar en una misma instrucción **LET** los dos tipos diferentes de variables, pues provocaríamos un error. A los dos lados del símbolo de igualdad deben existir siempre expresiones del mismo tipo:

110 LET T=T+C

En la línea 110, tanto la variable **T** como la variable **C** son numéricas, por lo tanto, la asignación por medio de **LET** es correcta. No lo sería, sin embargo:



La sentencia REM *se emplea para incluir comentarios en los programas.*

110 LET T$=T+C

Esta instrucción produciría un error, ya que la variable **T$** es de cadena (alfanumérica), y no podemos asignarle un valor numérico, como sería el resultado de la operación de suma entre las variables **T** y **C**.

CLS Y PRINT

La palabra clave **CLS** nos permite borrar el contenido de la pantalla. La siguiente instrucción

Los nombres de variables de cadena están compuestos por una letra seguida del símbolo dólar **($)**.

¡!

A una variable numérica nunca le podemos asignar resultados alfanuméricos, ni a una de este tipo un valor numérico.

*

Las únicas operaciones que se pueden efectuar con cadenas son la concatenación y el truncamiento.

*

El apóstrofe se utiliza en el Spectrum para forzar un «retorno de carro» en combinación con una instrucción **PRINT**.

Nunca se pueden realizar operaciones de igualdad entre expresiones de distinto tipo.

PRINT que coloquemos inmediatamente después de **CLS**, se situará al comienzo de la primera línea de la misma:

comprobar en el Ordenador

```
40 CLS
60 PRINT "Numero total:   ";N
```

En nuestro programa, el literal "Numero total: " aparecerá en la primera línea de la pantalla.
La instrucción **PRINT** es, sin duda, una de las más populares del vocabulario BASIC. Permite dar salida a la pantalla tanto de literales, que son transcritos en la forma exacta en que los incluimos en el programa, como de variables, o resultados de cálculos ejecutados por el ordenador:

```
160 PRINT "Nota media..:   ";M
```

Hubiera sido igualmente válido, ahorrándonos además el uso de la variable intermedia **M**, escribir:

```
160 PRINT "Nota media..:   ";T/N
```

Su uso más elemental se reduce a producir una línea en blanco en la pantalla, como es el caso de la línea 70 de nuestro programa:

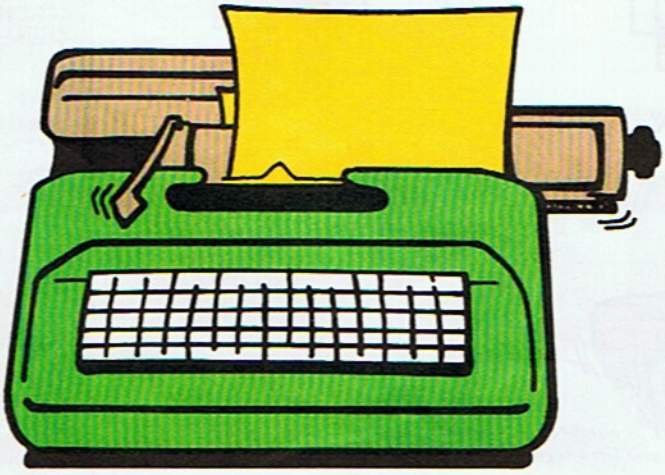
```
70 PRINT
```

Sin embargo, el empleo más potente y generalizado de la instrucción se da con los caracteres de puntuación BASIC.
Podemos imprimir una lista de variables, o mezcla de literales con variables indistintamente, en una misma línea separando cada elemento del que le sigue por medio del punto y coma (;):

```
160 PRINT "Numero total:   ";N;"Nota media..:   ";M
```

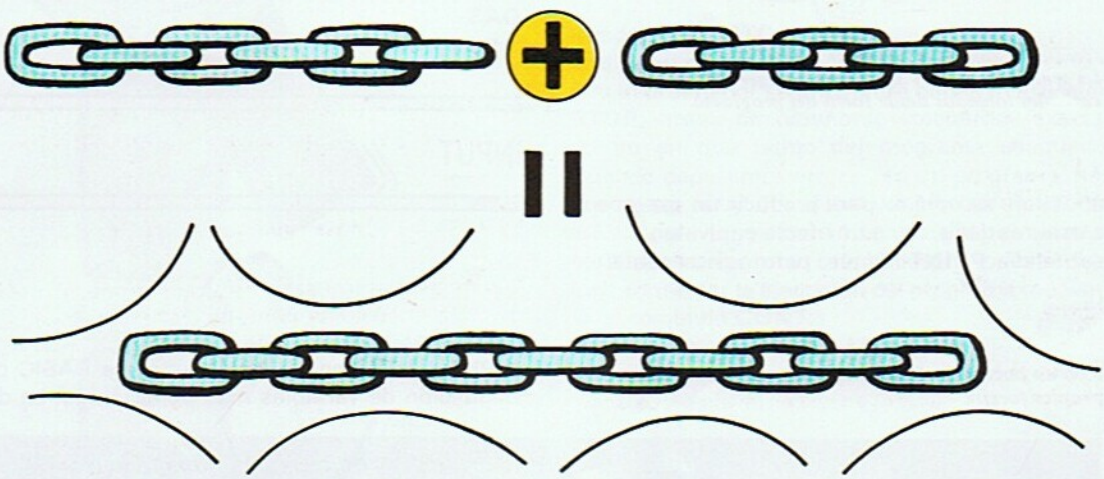
Si codificamos de esta manera la línea 160 de programa, debemos suprimir las instrucciones 60 y 70 que resultarían redundantes. De esta forma, al llegar el programa a la instrucción 160, se representarían en la pantalla, en una sola línea, los contenidos de las variables **N** y **M**, número de notas y nota media respectivamente, precedidas cada una de ellas por los literales descriptivos de su contenido.
Observamos que los literales terminan y comienzan, respectivamente, con un espacio en blanco. Esto es debido a que de no hacerlo así quedarían completamente pegados a las variables numéricas, efecto éste bastante antiestético.
De lo dicho deducimos que el ordenador cuando llega a esta instrucción, va efectuando la salida a la pantalla de los elementos que componen la línea, tanto los literales como las variables, unos

El apóstrofe tiene el efecto de un «retorno de carro» en combinación con una sentencia PRINT.



a continuación de otros, hasta que se detecta el final de la instrucción.
Como el último elemento a imprimir en nuestro ejemplo (la variable **M**), no va seguido de un punto y coma (;), el BASIC producirá automáticamente un «retorno de carro», similar al de una máquina de escribir eléctrica cuando pulsamos el cambio de línea, escribiéndose las siguientes sentencias **PRINT**, si existen, al comienzo de la siguiente línea de la pantalla.
Otro carácter de puntuación que empleamos en nuestro programa de ejemplo con la sentencia **PRINT** es la coma (,):

100 PRINT C,



La operación de concatenación consiste en la «suma» de cadenas.

Para entender el uso de este separador como carácter de tabulación, debemos suponer que el ancho de la pantalla (32 columnas) está dividido en dos zonas de 16. Si escribimos:

PRINT A,B

Obtenemos el valor de la variable **A** al comienzo de la primera mitad de la línea, y la variable **B** al comienzo de la segunda mitad de ésta. Hay que añadir que la coma (,) al final de una instrucción no produce un cambio de línea, por lo que, por sí sola, tiene un efecto equivalente a:

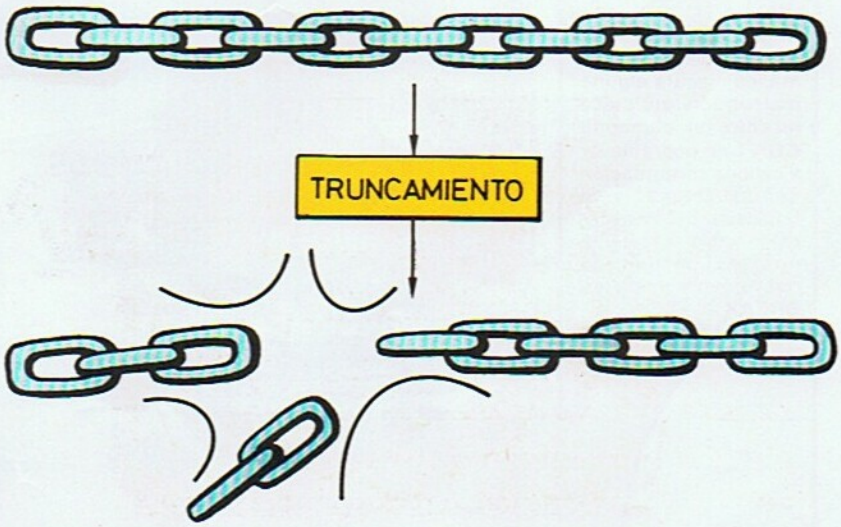
100 PRINT C,;

Por cada nuevo valor que suministramos a la variable **C** por medio de la instrucción **INPUT** de la línea 90, obtenemos la impresión de la variable en la siguiente «mitad de la pantalla» que le corresponda.
Tenemos como inconveniente que el número de calificaciones (**C**) puede ser impar y, en ese caso, la instrucción 140 se imprimiría en la segunda mitad de una línea de la pantalla, en contra de nuestros deseos. Para resolver ese problema, empleamos una sentencia **PRINT**, cuya única finalidad es que produzca un «retorno de carro» (cambio a la siguiente línea de la pantalla). Como además queremos que exista una línea en blanco, que separe las últimas calificaciones introducidas de la línea en que se imprime el resultado de la nota media, debemos incluir otra sentencia **PRINT**:

140 PRINT '

Esta instrucciíon nos da pie a introducir un último signo de puntuación que modifica los resultados de la sentencia **PRINT**: el apóstrofe. El

El truncamiento es la «extracción» de determinada parte de una cadena.



La interrupción de un programa durante la ejecución de un **INPUT** no se puede conseguir mediante **BREAK**, sino con **STOP**.

Para que el **STOP** sea admitido en un **INPUT** de variable de cadena, es necesario suprimir en primer lugar las comillas (") que caracterizan este tipo de datos.

Para continuar la ejecución de un programa que ha sido detenido mediante **BREAK** o **STOP** se utiliza el comando **CONT**.

Si después de la interrupción de un programa, se ejecuta alguna instrucción en modo directo, el comando **CONT** no podrá llevar a cabo la continuación del programa, y el Spectrum permanecerá «inactivo» hasta que no sea interrumpido nuevamente mediante **BREAK**.

La sentencia INPUT *sirve para recoger valores de variables tomados desde fuera del programa.*

apóstrofe se emplea para producir un «retorno de carro»; es decir, tiene un efecto equivalente a una sentencia **PRINT** simple, pero nunca puede ser empleado al comienzo de una instrucción, pues no se trata de una palabra clave BASIC. La utilización del apóstrofe nos permite ahorrar espacio; de no ser por él, para conseguir el efecto de dos retornos de carro en la línea 140, habríamos tenido que utilizar dos instrucciones en vez de una:

```
                    140 PRINT
140 PRINT ' =
                    140 PRINT
```

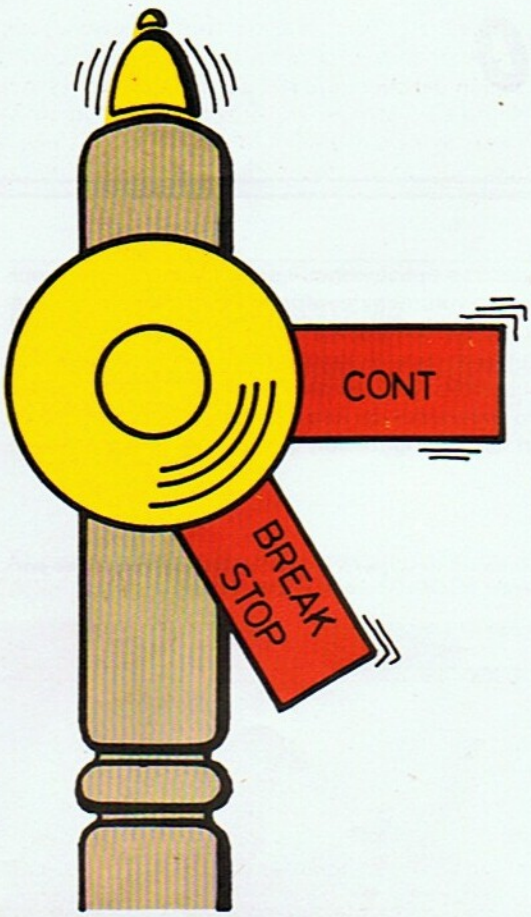
Y lo mismo habría sucedido en la línea 80:

```
80 PRINT "---CALIFICACIONES OBTENI
DAS---" '
                    =
80 PRINT "---CALIFICACIONES OBTENI
DAS---"
85 PRINT
```

INPUT

INPUT y **LET** son las palabras clave BASIC de asignación de variables más importantes. La di-

Con un poco de práctica llegaremos a realizar nuestros propios juegos.



Gracias al comando CONT *podemos continuar la ejecución de un programa que ha sido detenido mediante* BREAK *o* STOP.

ferencia fundamental entre ambas sentencias es que **INPUT** nos permite incluir el valor desde «fuera» del programa, decidiéndolo en el mismo momento de la ejecución, a petición del propio programa, que efectúa una parada para que se le suministre el valor que debe asignar a la variable. **INPUT** permite la introducción de variables numéricas:

50 INPUT "Numero total:";N

O variables de cadena:

170 INPUT "Fin de programa?";F$

Asimismo, el mensaje que hemos incluido de petición de variable en ambas líneas, es opcional y podemos eliminarlo escribiendo:

90 INPUT N o **170 INPUT F$**

Aparentemente no existe diferencia alguna entre que el **INPUT** corresponda a una variable numérica o de cadena. La diferencia la encontramos en el momento de la puesta en marcha del programa, ya que en la línea 170, por tratarse de una variable de *string* (cadena de caracteres), el Spectrum nos muestra el cursor **L** entre comillas, mientras que el **INPUT** numérico (línea 90) lo hace sin ellas.

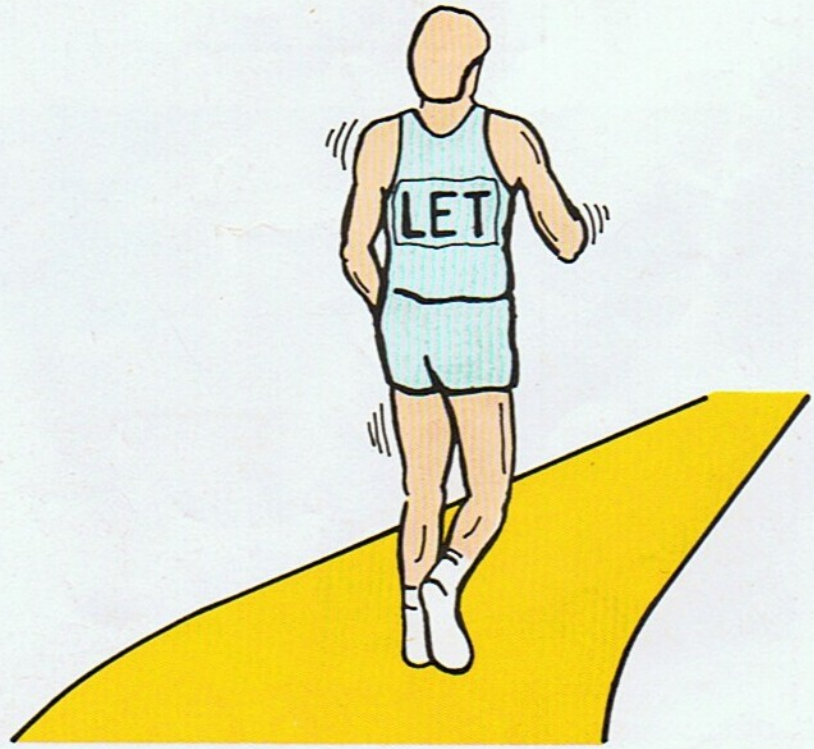
Si deseamos interrumpir el programa cuando nos encontramos dentro de una sentencia **INPUT**, podemos hacerlo por medio de la tecla **STOP** en lugar de **BREAK.** En el caso de que el **INPUT** sea numérico, **STOP** funcionará sin más dificultades, pero si se trata de una variable de cadena, debemos eliminar previamente las comillas avanzando una posición el cursor y pulsando dos veces **DELETE** para, finalmente, pulsar **STOP** y **ENTER.**

Desde luego **STOP** no implica necesariamente una detención definitiva del programa. Siempre que el ordenador se encuentra con la instrucción **STOP,** antes de detenerse «recuerda» exactamente en qué punto del programa se quedó. Cuando depuramos errores en un programa, nos puede ser muy útil interrumpirlo por medio de **STOP** para preguntar el valor de determinadas variables y comprobar resultados; después podremos continuar la ejecución del programa mediante el comando **CONT.**

¡!

Para borrar toda la línea tecleada en un **INPUT,** antes de la pulsación de **ENTER,** se puede emplear la tecla **EDIT (CAPS SHIFT + 1).** En caso de **INPUTs** de cadenas, las comillas que los caracterizan también desaparecerán por este sistema.

La sentencia LET *asigna valores a las variables desde dentro del programa.*



INVESDISK 200

QUÁNTAS veces todos aquellos que manejamos habitualmente el Spectrum, nos desesperamos cuando al grabar o cargar un programa almacenado en cinta, tenemos que aguardar varios minutos hasta completar la tarea; es más, puede que recibamos con cara de pocos amigos la noticia de que ese programa que tanto nos costó realizar, ha quedado reducido a un exasperante mensaje en la parte inferior de la pantalla: **R Tape loading error.**

Por desgracia, estos problemas se suceden con una relativa frecuencia. Desde luego, las cintas tienen sus ventajas; son baratas y como medio de almacenamiento masivo de datos y programas son una buena solución. Pero la transmisión de la información es lenta, y para un tratamiento «profesional» de datos están bastante lejos de ser el medio ideal.

La unidad de disco comercializada por INVESTRONICA, viene a completar la amplia gama de periféricos disponibles para el Spectrum. Desde luego, si nuestro ordenador tan sólo nos sirve como base de lanzamiento de misiles, los discos estarán de más; pero si lo que necesitamos en nuestras aplicaciones es manejar con gran rapidez un volumen de información considerable, disponer de ficheros de acceso aleatorio, y una garantía de que lo almacenado no se convertirá en nubes de humo, el INVESDISK 200 dotará a nuestro ordenador de una capacidad profesional que, en su configuración básica, le estaba vetada.

Los casetes son el medio de almacenamiento de la información, de más bajo coste.

El sistema de disco INVESDISK 200 está compuesto por cuatro módulos de color negro.

DESCRIPCION DEL SISTEMA

El sistema consta de cuatro módulos del mismo color que el ordenador: INTERFACE, ALIMENTADOR DE CORRIENTE, CONTROLADOR y UNIDAD DE DISCOS.

El interface (el más pequeño de los cuatro elementos), se conecta por una parte a la tarjeta de expansión del Spectrum, y por otra al controlador, mediante un conector del tipo D con 15 terminales. En su parte superior, incorpora un botón de *reset,* que permite en cualquier momento restaurar la situación inicial del ordenador (con la consiguiente pérdida de la información de la memoria).

El objetivo del alimentador es suministrar la co-

¡!

El controlador es un elemento fundamental en el sistema INVESDISK 200; en él reside el sistema operativo TOS (Timex Operating System), y es el encargado de controlar todas las operaciones relacionadas con el disco. Para llevar a cabo su tarea se auxilia en un microprocesador Z-80 y una memoria interna independiente de la del Spectrum.

*

Los disquetes utilizados por INVESDISK 200 poseen dos tipos de protecciones: una contra el borrado y escritura accidental por el usuario, consistente en unos obturadores deslizantes (uno por cada cara); y otra contra el deterioro físico del soporte magnético, que se encuentra protegido por una placa metálica que impide su exposición al medio ambiente, y encerrado en una caja rígida para evitar que se dañe al ser doblado.

¡!

La cabeza de lectura/escritura de la unidad de disco fundiona por una sola cara (la superior del disquete que se inserta), debido a ello no se pueden acceder simultáneamente más que a 160 K de las 320 del disquete.

*

El sistema INVESDISK 200 es incompatible con el ZX INTERFACE 1, y por tanto, con los Microdrives, interface RS-232 y red de área local.

rriente necesaria para el perfecto funcionamiento del sistema. Un interruptor en su parte posterior, marcado con 0-1, permite activarlo. De esta unidad parten cuatro cables; uno para la conexión a la red, otro que la une al controlador y finalmente, dos cables para las unidades de disco. Si sólo disponemos de un *drive* (unidad de disco), uno de estos cables quedará libre. Por el contrario, si deseáramos conectar tres o cuatro unidades, necesitaríamos una segunda fuente de alimentación.

El controlador tiene la misión de procesar todas las instrucciones que recibe desde el ordenador con destino a las unidades de disco. Este elemento del sistema INVESDISK puede ser considerado como un «periférico inteligente», ya que tiene su propio microprocesador Z-80 y su memoria de acceso independiente, por lo que se puede decir que

El CONTROLADOR posee su propio microprocesador Z-80 para tratar los datos procedentes del ordenador.

es una reducidísima versión de un microordenador, cuya única función es controlar las unidades de disco.

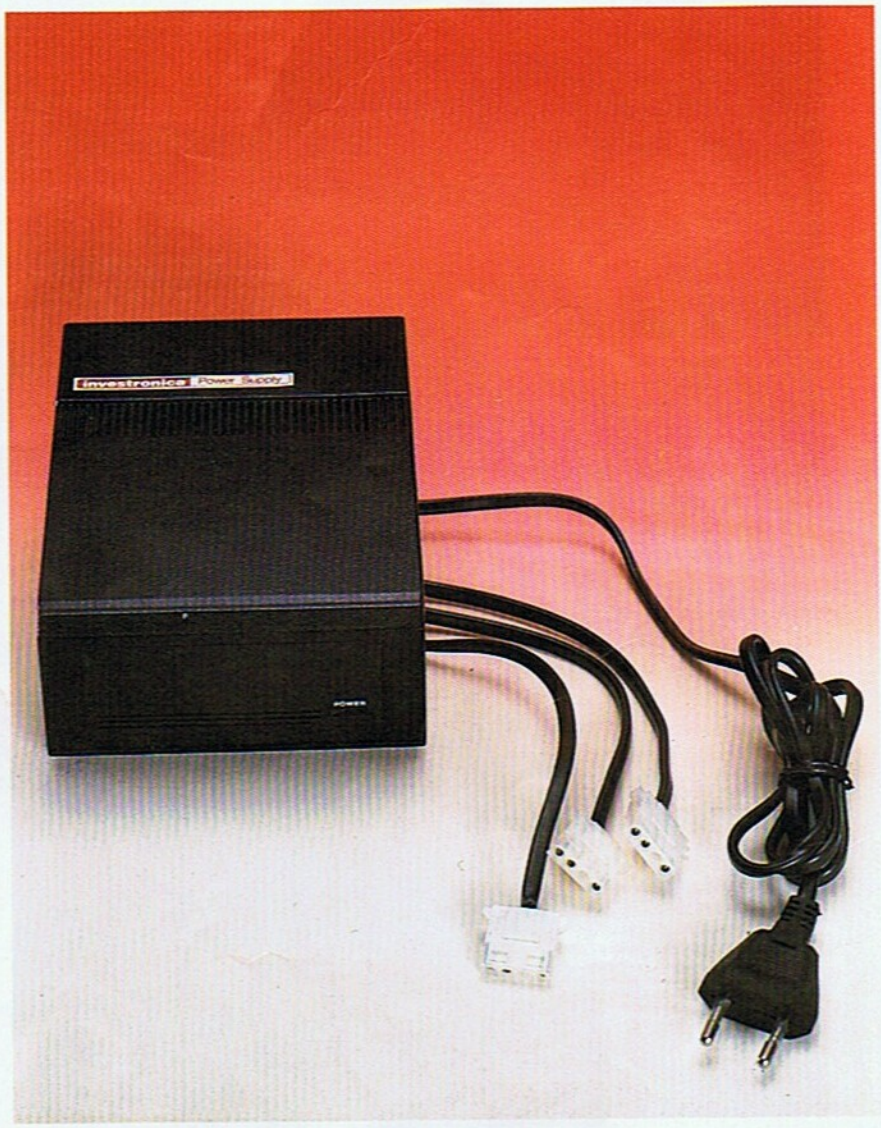
Cada vez que el controlador recibe una orden, la almacena en su memoria interna, se encarga de ejecutarla y, además, no ocupa memoria del Spectrum. Es en este módulo donde reside el TOS (Timex Operating System), sistema operativo para

El INTERFACE es el más pequeño de los cuatro componentes del equipo.

disco de la firma TIMEX, necesario para realizar cualquier tarea que se le encomiende al disco. Un pulsador de *reset* permite el borrado de la memoria interna de la unidad, y la posibilidad de que, si en el disquete insertado en la unidad existe un programa con el nombre START, éste se cargue y ejecute automáticamente.
El último módulo del sistema es la unidad de disco. En su frontal se encuentra situada la ranura por donde introducir el disquete. Un indicador luminoso de color rojo, se enciende cada vez que el disco entra en funcionamiento. Si es la primera ocasión en que se conecta el disco, o hemos realizado un *reset* en el controlador, el citado indicador permanecerá parpadeante hasta completar la carga del sistema operativo (TOS). Si ahora efectuamos un *reset* en el interface, además del habitual mensaje de presentación de Sinclair, obtenemos en la pantalla la inscripción: **(C) 1984 TIMEX-TOS VA.1.**

LOS DISQUETES

Los disquetes utilizados son de tres pulgadas, con sus dos caras disponibles para nuestro uso. La capacidad de almacenamiento de información por cada cara es de 160 K, aunque en realidad, tras la inevitable operación **FORMAT** que más adelante estudiaremos, no quedarán más que 140 Kbytes aprovechables. Esto se debe a que el sistema operativo ocupa aproximadamente 20 K, y es del todo necesario que se encuentre grabado en el disquete, antes de poder realizar cualquier operación con él.
Los disquetes disponen físicamente de dos protecciones contra el posible deterioro de la información almacenada en ellos. La primera de ellas está constituida por dos pequeños dispositivos deslizantes de plástico rojo; según la posición en que éstos se encuentren, la cara del disquete a que afectan podrá ser sometida o no a operaciones de escritura o borrado, quedando siempre abierta la posibilidad de lectura.
Este mecanismo es similar al de los casetes o cartuchos para microdrives Sinclair, con la diferencia de que en este caso su efecto es fácilmente reversible: basta con deslizar nuevamente el obturador plástico en sentido contrario, sin tener necesidad de «parchear» el disquete con cinta adhesiva u otros medios, como sucede con los cartuchos y los casetes, e incluso en otros tipos de disquetes.
La segunda protección que poseen estos *floppies* (nombre por el que se conoce a los disquetes con soporte magnético flexible), no es ya contra el borrado accidental, sino contra el posible deterioro físico que pueda sufrir el dispositivo. La plancha de material magnético está encerrada en una caja rígida y la zona en la que se sitúa la cabeza de lectura/escritura del disco, se encuentra protegida por una placa metálica que evita la exposición directa de la superficie magnética al medio ambiente. Gracias a estas características se evitan los dos problemas más frecuentes con este tipo de soportes de información: doblar el disquete o tocar con los dedos la superficie magnética. El sistema por el cual el disquete queda expuesto a la acción de la cabeza de lectura/escritura



Al contrario que el ZX Microdrive, el disco distribuido por INVESTRONICA precisa de una fuente de alimentación independiente de la del Spectrum.

El disquete empleado por INVESDISK 200 es un floppy de tres pulgadas, utilizable por las dos caras, con una capacidad de 160 K (antes del «formateado») por cada una de ellas. Para la protección del disquete contra la escritura se dispone de dos pequeños elementos deslizantes de color rojo, que se encuentran ubicados en la parte delantera del mismo. Cuando el disquete se inserta en la unidad de disco, la placa metálica que habitualmente protege la superficie magnética se desplaza para permitir las operaciones de lectura y escritura.

del disco es bastante ingenioso: una palanca de plástico blanco situada en un costado del mismo, se encuentra unida a la placa metálica de protección antes mencionada; conforme el disquete se va insertando en la unidad de disco, la palanca se va desplazando, dejando al descubierto la superficie magnética para que pueda ser accedida libremente por la cabeza del disco.

Aunque antes hemos visto que la capacidad de cada disquete es de 160 K por cada cara (140 K útiles), no hemos de pensar que disponemos de 320 K por disquete, puesto que la cabeza de la unidad de disco no puede trabajar por las dos caras. Esto quiere decir que adquirir un disquete con destino al INVESDISK 200, es como comprar a un mismo tiempo 2 disquetes de 160 K, pero no debemos engañarnos pensando que podemos acceder, sin darle la vuelta al disquete, a sus 320 K de información. Debido a esto, cuando se realiza una operación **FORMAT** para preparar el disquete virgen para su uso, ésta no se realiza sobre las dos caras.

En todo caso, su capacidad de almacenamiento es bastante considerable (casi el doble que la de un cartucho de microdrive), aunque su punto fuerte se encuentra en las posibilidades de manejo del disco, más que en su capacidad de almacenamiento de información.

CARACTERISTICAS DEL DISCO

Al igual que el ZX INTERFACE 1, el sistema operativo TOS utiliza el símbolo asterisco (*) para identificar los comandos destinados al disco. Así, palabras BASIC como **GOSUB, ATTR** o **MOVE,** adquieren un significado adicional al acoplar el sistema INVESDISK 200 a nuestro ordenador.

La unidad de disco se acompaña de un disquete con programas de demostración.

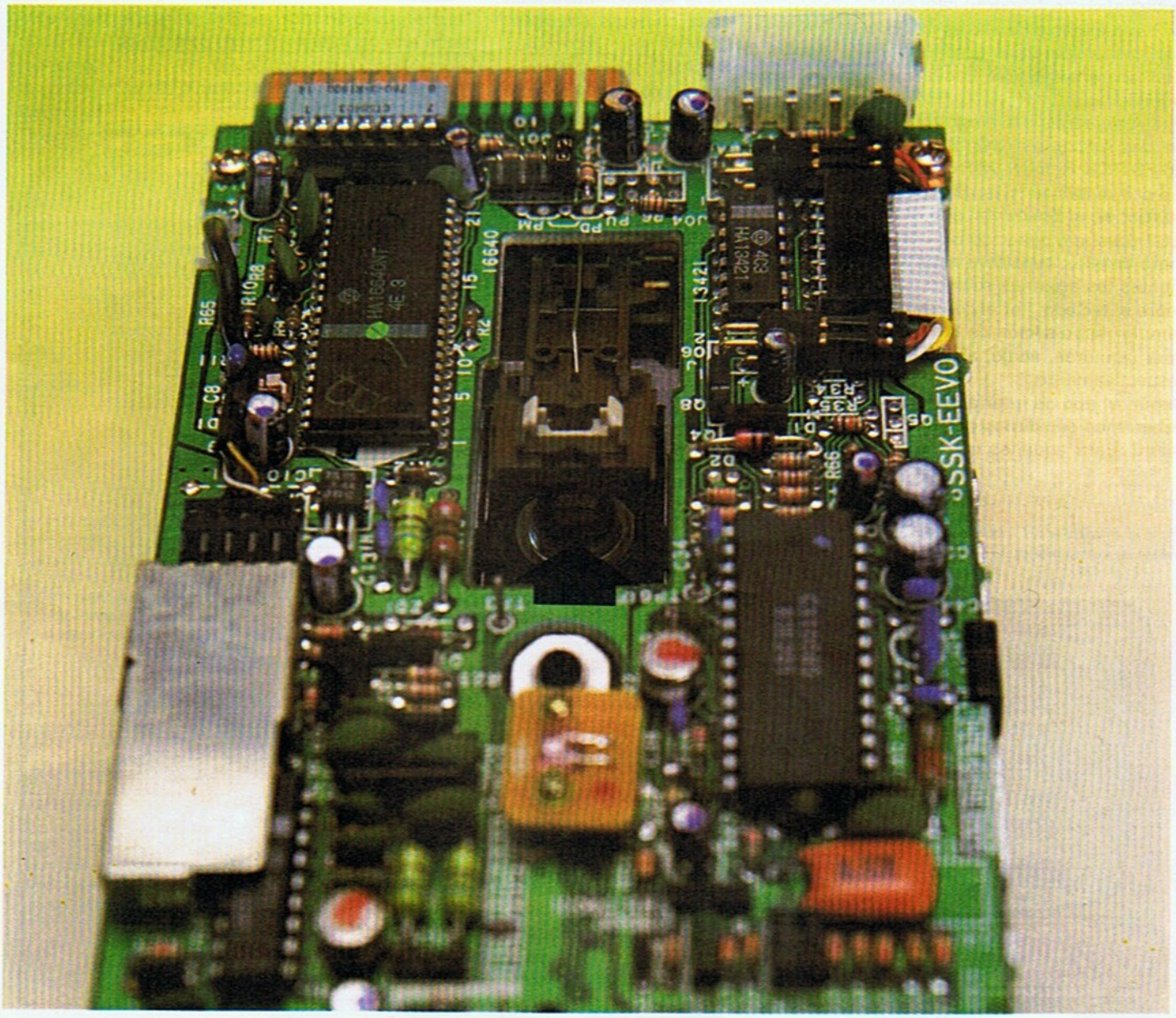
Desde luego, el manejo del casete no es en ningún momento entorpecido por el disco, gracias a lo cual podremos disponer simultáneamente de dos tipos diferentes de almacenamiento de datos: el disco para el trabajo diario que precisa de rapidez y agilidad en el tratamiento, y el casete para las baratas y siempre necesarias copias de seguridad **(BACK-UP)**.

Quizás el único inconveniente que podamos encontrar en la incompatibilidad INTERFACE 1-INVESDISK es que nos priva de dos herramientas asociadas al uso del interface de Sinclair: el interface serie RS-232 y la interconexión en red de área local **(NET)**.

Como último dato añadiremos que el sistema operativo pone a nuestra disposición dieciséis canales diferentes por los que circularán los datos. Los canales del 1 al 4 son rápidos, y del 5 al 16, lentos. Esto se debe a que cada canal rápido tiene reservado un *buffer* de 512 bytes exclusivamente para él. Los lentos comparten un único *buffer* común, de forma que cada vez que entra una nueva información en él, se destruye la que hubiera almacenada con anterioridad.

Unas páginas más adelante finalizaremos el examen del INVESDISK 200, estudiando más a fondo la capacidad y características de su sistema operativo.

En la zona central de la foto podemos apreciar la cabeza de lectura/escritura de la unidad de disco. Al introducir un disquete, la superficie magnética queda automáticamente expuesta a la acción de la cabeza de lectura/escritura del disco.



TELE-SKETCH

EL objetivo de este programa no es otro que el de acercarnos al empleo del *joystick* en nuestros propios programas, con el fin de que la palanca de juego no se convierta en un instrumento de poca utilidad, que sólo empleamos para abatir naves interestelares o masacrar marcianitos. La utilización de este periférico ayuda en muchas ocasiones al control de gran cantidad de programas. Un buen ejemplo de ello es el **TELE-SKETCH** del que nos ocuparemos a continuación, gracias al cual podremos trazar líneas en la pantalla del ordenador.

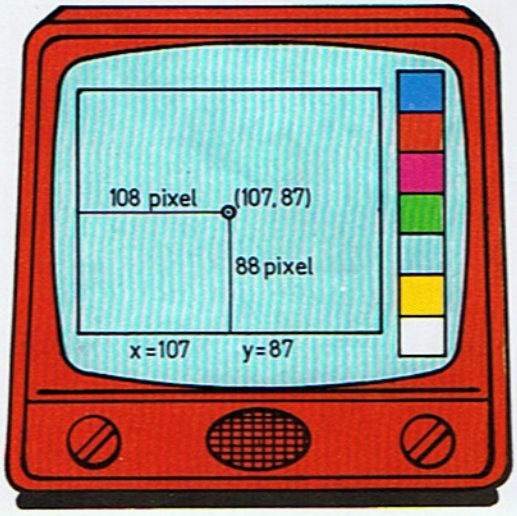
Una vez ejecutado el programa, aparece un menú de dos opciones: **JOYSTICK KEMPSTON** y **JOYSTICK & CURSORES.** Como ya sabemos, los dos sistemas más difundidos de control del *joystick* en el Spectrum, son el **KEMPSTON** y el **AGF** (también conocido como **CURSOR**). El primero de ellos, considera absolutamente independientes las acciones sobre la palanca de juego y sobre el teclado. Por el contrario, el sistema **AGF** «emula» la pulsación de las teclas 5, 6, 7, 8 y 0, para izquierda, abajo, arriba, derecha y disparo, respectivamente.

Según el tipo de interface del que dispongamos, deberemos decidirnos por una u otra opción del menú. Para aquellos que no poseamos un *joystick*, el segundo punto del menú nos permitirá manejar el programa en virtud a las teclas 5, 6, 7, 8 y 0, antes mencionadas. En todo caso, hay que insistir en que este programa no es importante por el trabajo que realiza, sino porque sirve como ejemplo de la forma de controlar la palanca de juego en nuestros propios programas, por lo que será de mayor interés el estudio del mismo, que su utilización.

El resultado de la lectura del PORT *empleado por el interface* KEMPSTON (IN 31), *debe ser interpretado según los datos expuestos en el gráfico.*

Una vez que hayamos seleccionado una opción del menú, aparecerá en el margen izquierdo de la pantalla una zona de dibujo acotada por un rectángulo blanco, y a su derecha, una representación de los siete colores que podemos emplear (el fondo de la pantalla es negro). En la parte inferior, se imprimen las coordenadas del punto de dibujo: **X** para el eje horizontal e **Y** para el vertical.

Para el comienzo del trazado de la línea se ha escogido el punto de coordenadas 107,87.



MANEJO DEL PROGRAMA

Para trazar cualquier línea, sólo tenemos que utilizar el *joystick* o el teclado. Cuando deseemos cambiar el color de escritura, pulsaremos el botón de disparo (o la tecla **0** si hemos elegido la opción de teclado). Acto seguido, una punta de flecha recorrerá cíclicamente el menú de colores que se encuentra a la derecha de la pantalla; cuando el indicador apunte al color que deseamos utilizar, hemos de pulsar nuevamente el botón de disparo, y la tinta quedará fijada para los próximos trazados.

¡!

Al analizar la lectura del **PORT** 31, controlado por un interface **KEMPSTON**, debemos tener en cuenta que la lógica seguida es positiva, es decir, las acciones sobre la palanca colocan a 1 los bits correspondientes.

*

La lógica seguida por los sistemas **AGF**, **PROTEK** y **CURSOR**, así como por el teclado, es negativa. Por tanto, la pulsación de una tecla, o la acción sobre el **joystick**, sitúan a 0 los bits correspondientes.

Cuando se selecciona un nuevo color, nuestra posición actual se destaca por un círculo blanco que a continuación desaparece, permitiéndonos continuar el trazado del dibujo. Finalmente, podremos utilizar la tinta negra, igual a la de fondo, para «hacer invisibles» determinadas líneas del gráfico. Esto lo conseguiremos pulsando la tecla **B.**

Sólo nos queda recordar que, como ya estudiamos anteriormente, el Spectrum no permite que dos líneas de distinto color coexistan en una misma posición de carácter. Al intentarlo, podremos apreciar cómo la línea trazada con anterioridad adopta el color de la recién llegada.

Una vez introducido el programa, podemos grabarlo en cinta mediante **SAVE "JOY" LINE 1,** si deseamos que se autoejecute al cargarlo; o bien **SAVE "JOY",** para que se realice simplemente la carga.

Para conseguir interpretar el estado del **joystick** con el interface **KEMPSTON,** es necesario realizar la lectura del **PORT** 31 (decimal), lo cual se consigue mediante el comando BASIC **IN 31.** El resultado obtenido debe interpretarse de la siguiente manera: DERECHA = 1, IZQUIERDA = 2, ABAJO = 4, ARRIBA = 8 y DISPARO = 16. Si se desean estudiar resultados combinados, debemos descomponer el movimiento en sus sentidos principales, y sumarlos. Por ejemplo, la diagonal

superior derecha implica el resultado 9, puesto que el desplazamiento hacia arriba produce un valor 8 y el de la derecha un valor 1.

Si obtenemos la forma binaria del resultado, su análisis será mucho más sencillo. De los ocho bits que componen el byte, los tres bits de la izquierda no nos deben importar, ya que sólo son posibles cinco estados simples (ARRIBA, ABAJO, IZQUIERDA, DERECHA y DISPARO). De los cinco bits restantes (los de la derecha), cada uno corresponde a uno de los estados. Concretamente, interpretándolos de derecha a izquierda: DERECHA, IZQUIERDA, ABAJO, ARRIBA y DISPARO. Si el bit estudiado está a 1, indica que el movimiento correspondiente ha sido realizado, en caso contrario, estará a 0.

Veamos un ejemplo. Si el jugador ha realizado un movimiento en la diagonal superior derecha, y al mismo tiempo ha pulsado el botón de disparo, los bits 0, 3 y 4 (contando de derecha a izquierda y empezando por el cero), estarán a 1 (lógicamente, el resto serán ceros). Por tanto, el byte tendría esta forma en binario: 000 11001, lo cual traducido a decimal supone un valor 25 (**PRINT BIN 00011001**). Efectivamente, descomponiendo el movimiento en sus sentidos principales, obtendríamos: 1 (ARRIBA) + 8 (DERECHA) + 16 (DISPARO) = 25.

Si el interface empleado es el de los cursores (**AGF** o **PROTEK**), podremos seguir procedimientos parecidos a los anteriores. Para estudiar resultados simples, es decir, los que no son combinación de varias acciones, utilizaremos la función BASIC **INKEY$**, de la siguiente forma: DERECHA = "5", ABAJO = "6", ARRIBA = "7", DERECHA = "8" y DISPARO = "0".

Por el contrario, para analizar movimientos compuestos, deberemos recurrir en esta ocasión a estudiar las formas binarias de dos direcciones del **PORT**. El bit 4 del **IN 63486**, indicará el estado de la tecla "5" (DERECHA) y los bits 0, 2, 3 y 4 (contando a partir de cero, y de derecha a izquierda) del **IN 61438**, responderán a los movimientos siguientes: DISPARO, DERECHA, ARRIBA y ABAJO, respectivamente. Para terminar, hemos de destacar algo muy importante: cuando se realizan lecturas del teclado, los bits a 1 indican teclas no pulsadas, y los bits a 0, teclas pulsadas; es decir, la lógica inversa a la utilizada con el interface **KEMPSTON**.

```
  5 REM ************************
  6 REM * J.M.MAYORAL SERRANO *
  7 REM ************************
  8 GO SUB 2090
  9 POKE 23658,8: LET K=9: PAPER 0: INK 9: BORDER 0:
CLS
 10 FOR P=0 TO 6
 20 FOR N=0 TO 2
 30 PRINT  PAPER P+1;AT P*3+N,28;"
 40 NEXT N
 50 NEXT P
 80 FOR Y=175 TO 7 STEP -24
 90 PLOT 255-33,Y
100 DRAW 27,0
110 NEXT Y
120 FOR X=255-33 TO 255 STEP 27
130 PLOT X,8
140 DRAW 0,166
150 NEXT X
160 DATA 215,0,0,153,-215,0,0,-153,74,46,77,46,77,65
,89,79,82,65,76,32,32,49,57,56,53
170 PLOT 0,12
180 FOR N=1 TO 4
190 READ A,B
200 DRAW A,B
210 NEXT N
220 FOR N=1 TO 17
225 READ C
230 PRINT AT 0,4+N;CHR$ C
240 NEXT N
250 REM
300 REM MOVIMIENTO CURSOR
305 REM
310 LET X=107: LET Y=87
320 PLOT X,Y
330 PRINT AT 21,5;"X=";X;"  ","Y=";Y;"  "
335 IF E$="2" THEN  GO TO 2180
340 LET I=IN 31
342 IF INKEY$="B" THEN  LET K=0
343 IF I>=16 THEN  GO SUB 1000
345 IF I=0 THEN  LET X=PEEK 23677: LET Y=PEEK 23678:
GO TO 340
350 LET R=LN I/LN 2
360 IF R-INT R<>0 THEN  GO TO 340
370 GO SUB R*100+400
380 GO TO 340
395 REM DERECHA
403 IF X=213 THEN  LET X=X-1: RETURN
405 PLOT  INK K;X,Y
410 LET X=X+1
415 GO SUB 900
420 RETURN
495 REM IZQUIERDA
500 IF X=2 THEN  RETURN
505 PLOT  INK K;X,Y
510 LET X=X-1
515 GO SUB 900
520 RETURN
595 REM ABAJO
600 IF Y=14 THEN  RETURN
605 PLOT  INK K;X,Y
610 LET Y=Y-1
615 GO SUB 900
620 RETURN
695 REM ARRIBA
700 IF Y=163 THEN  RETURN
705 PLOT  INK K;X,Y
710 LET Y=Y+1
715 GO SUB 900
720 RETURN
795 REM PINTA
900 PRINT AT 21,7;X;"  "
910 PRINT AT 21,18;Y;"  "
920 RETURN
1000 REM COLOR
1020 LET F=1
1030 PRINT  OVER 1;AT F,31;"<"
1080 BEEP .2,20: BEEP .3,21
1090 PRINT  OVER 1;AT F,31;"<"
1091 IF E$="1" THEN  GO TO 1095
1093 IF INKEY$="0" THEN  GO TO 2000
1094 GO TO 1100
1095 IF IN 31>=16 THEN  GO TO 2000
1100 LET F=F+3
1105 IF F>=20 THEN  LET F=1
1110 GO TO 1030
1900 DATA 1,1,2,4,3,7,4,10,5,13,6,16,7,19
2000 RESTORE 1900
2010 FOR N=1 TO 7
2020 READ A,B
2030 IF F=B THEN  LET K=A
2040 NEXT N
2050 FOR N=1 TO 4
2055 CIRCLE  OVER 1;X,Y,1
2070 NEXT N
2080 RETURN
2090 REM MENU
2100 FOR N=1 TO 2
2110 PRINT  FLASH 1;AT N*7,3;N
2120 NEXT N
2130 PRINT AT 7,5;"JOYSTICK KEMPSTON"
2140 PRINT AT 14,5;"JOYSTICK & CURSORES"
2150 LET E$=INKEY$
2160 IF CODE E$<49 OR CODE E$>50 THEN  GO TO 2150
2170 RETURN
2180 REM J.CURSOR
2190 IF INKEY$="5" THEN  GO SUB 500
2200 IF INKEY$="6" THEN  GO SUB 600
2210 IF INKEY$="7" THEN  GO SUB 700
2220 IF INKEY$="8" THEN  GO SUB 400
2230 IF INKEY$="0" THEN  GO SUB 1000
2240 IF INKEY$="B" THEN  LET K=0: GO TO 2190
2250 GO TO 2185
```

¡!

Para el control del trazado de la línea, el programa utiliza la variable del sistema **COORDS**. La dirección decimal 23677, contiene la coordenada **X** del último punto trazado, y 23678 la coordenada **Y** del mismo.

*

El sistema **KEMPSTON** considera independientes las acciones sobre el teclado y el *joystick*. Por el contrario, los sistemas **AGF**, **PROTEK** y **CURSOR**, emulan la pulsación de las teclas 5, 6, 7, 8 y 0, para las acciones sobre el *joystick*.

*

El sistema **KEMPSTON** emplea una sola dirección del **PORT**, la 31 (decimal). Por el contrario, los emuladores de los cursores, precisan de la lectura de dos direcciones: la 63486 y la 61438.

En el análisis de la fila superior del teclado, el bit 4 del IN 63486 corresponde a la tecla "5"; los bits 0, 2, 3 y 4 del IN 61438, se ven afectados por la pulsación de las teclas "0", "8", "7" y "6", respectivamente.

EL SPECTRUM DECIDE

A través de los programas propuestos hasta ahora, hemos podido comprobar que el ordenador ejecuta siempre las instrucciones que componen los mismos de una manera secuencial, es decir, una a continuación de otra, respetando su numeración. Este estado de cosas sólo puede ser modificado mediante la sentencia **GO TO**, que obliga a un cambio en la siguiente línea a ejecutar en el programa.

Vamos a ver ahora como nuestro Spectrum es realmente capaz de «tomar decisiones» en base a comparaciones, y seguir por un camino u otro a partir de una evaluación de condiciones. La estructura general de la instrucción a tal fin es:

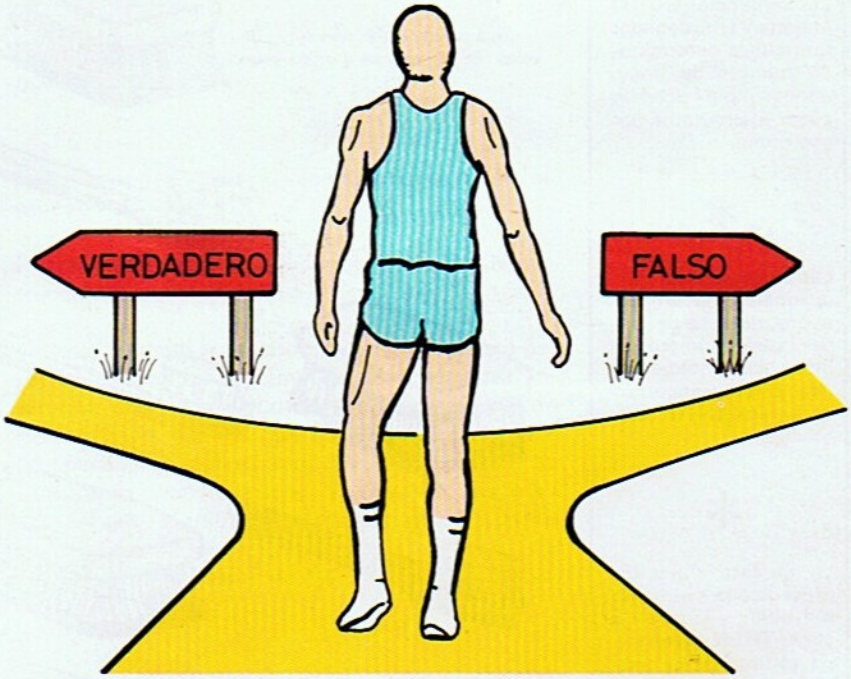
IF condición **THEN** instrucción

IF, que en inglés quiere decir SI (si condicional), precede a la condición a evaluar, y **THEN**, cuya traducción es ENTONCES, sirve para determinar el final de la condición e indicar el comienzo de la instrucción, que ha de ejecutarse sólo en caso de que la evaluación haya sido dada por «verdadera». Si, por el contrario, la evaluación de la condición resulta «falsa», pasa a ejecutarse la siguiente línea, haciéndose caso omiso de la instrucción que sigue a **THEN**. El comportamiento de este tipo de instrucciones podríamos describirlo llanamente de la siguiente manera:

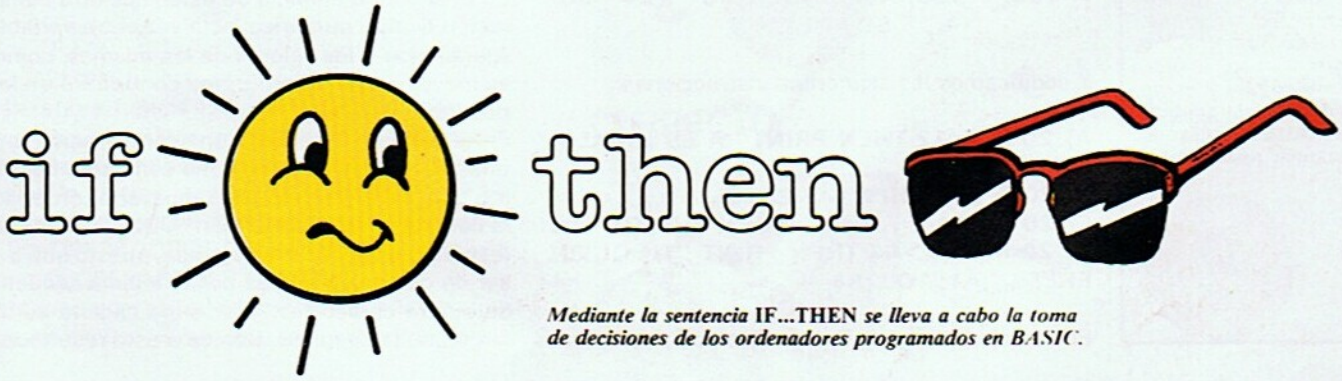
Si se cumple determinada condición ENTONCES haz lo que sigue SINO...

Podemos establecer comparaciones de tipo numérico o de cadena, pero nunca mixtas, es decir, comparar letras con números. Dados dos elementos, podemos realizar las siguientes comparaciones: si son iguales, si el primero es menor que el segundo, si el primero es mayor que el segundo, si el primero es menor o igual que el segundo (no es mayor), si el primero es mayor o igual que el segundo (no es menor), y si el primero es diferente del segundo (no es igual). Con esto se agotan todas las combinaciones posibles. Los símbolos que corresponden a estas comparaciones son:

=, <, >, <=, >= y <>



Las sentencias IF *obligan al Spectrum a tomar una decisión entre dos posibles opciones.*



Mediante la sentencia IF...THEN *se lleva a cabo la toma de decisiones de los ordenadores programados en BASIC.*

¡!

Las sentencias **GO TO** imponen al ordenador un salto a determinado número de línea, por el simple hecho de pasar el programa por ese punto.

*

Las bifurcaciones condicionales, hacen que el programa se dirija a un nuevo número de instrucción, sólo en el caso de que concurran determinadas circunstancias.

*

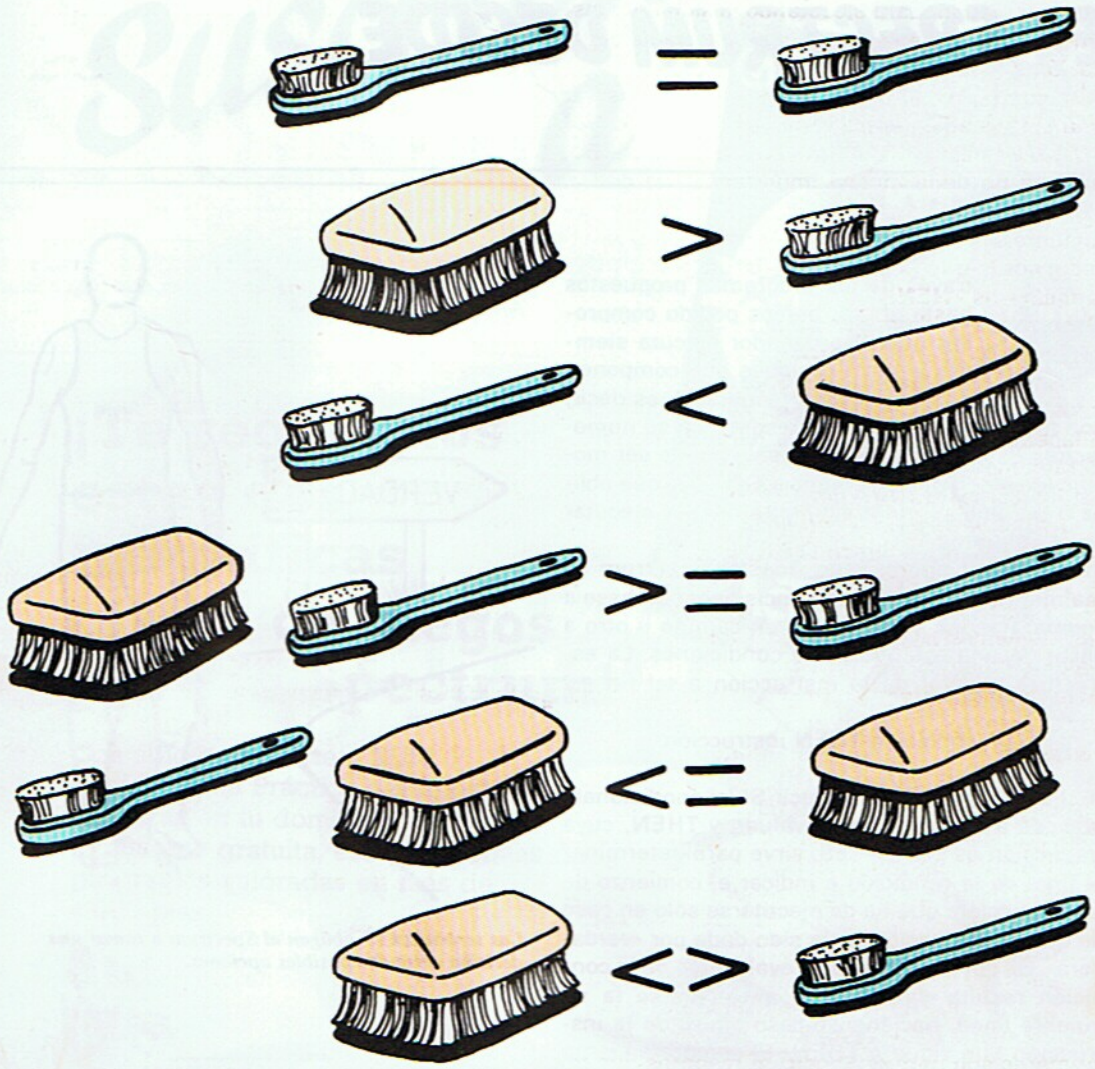
La palabra clave **IF**, precede a la expresión a evaluar, y la palabra clave **THEN**, determina el final de la condición y el principio del comando a ejecutar, si la evaluación de las condiciones resulta verdadera.

*

Las comparaciones deben establecerse siempre entre variables del mismo tipo, es decir, dos numéricas o dos de cadena.

*

Se puede continuar siempre el programa, a pesar de que no haya transcurrido la totalidad del tiempo asignado a la sentencia **PAUSE**, pulsando cualquier tecla.



Son seis las posibles comparaciones lógicas que se pueden realizar en el BASIC de Sinclair.

Como dicho así resulta un poco árido, aplicaremos nuestros conocimientos a algunos ejemplos. Vamos a suponer que, dentro de un programa, los valores de cuatro variables son:

A=162, B=165, A$="JOSE LUIS" y B$="JOSE ANGEL"

Y codificamos las siguientes instrucciones:

A) **20 IF A=12 THEN PRINT "A ES IGUAL A 12"**
B) **20 IF B>A THEN GO TO 90**
C) **20 IF A$<="JOSE" THEN GO TO 100**
D) **20 IF B$<>A$ THEN PRINT "TU QUIEN ERES? ";A$;" O ";B$**

En los ejemplos A y B, se establecen comparaciones entre elementos de tipo numérico, y en C y D de tipo cadena.

En el primer caso, se compara la variable **A** con la constante numérica **12.** Como la variable **A** tiene valor 162 en el momento de acceder el programa a la línea 20, no se ejecutará la sentencia **PRINT** que figura a partir de **THEN,** sino que el programa continuará en secuencia con la línea siguiente a la instrucción 20.

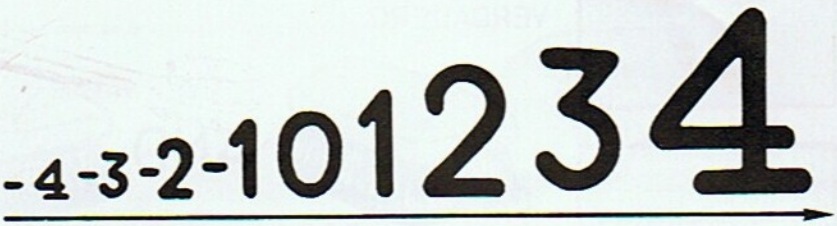
En el segundo ejemplo se establece otra comparación de tipo numérico, esta vez entre variables. Atendiendo a los valores de las mismas, como **B** es mayor que **A,** el programa continuará en la línea 90.

En el ejemplo C aparece una comparación entre una variable de cadena y una constante del mismo tipo. En este caso, al comparar el ordenador la cadena "JOSE LUIS" con "JOSE", la primera le resulta mayor que la segunda, puesto que a pesar de comenzar ambas por la misma secuencia de cuatro caracteres, la primera cadena es más larga que la segunda. Debido a este resultado, el

programa continuará ejecutando la siguiente instrucción en secuencia, y no la sentencia de bifurcación a la línea 100.
Finalmente, en el último ejemplo, podemos ver una comparación entre variables de cadena (**A$** y **B$**). En este caso, **A$** resulta mayor que **B$**, aunque no tiene mucha importancia, ya que lo que tratamos de averiguar en esta ocasión es solamente si las variables son diferentes en contenido, por lo que el programa ejecutará el **PRINT** situado tras **THEN.**
Debemos considerar que cuando tratamos variables de cadena en comparaciones, se utiliza el ordenamiento interno que el Spectrum tiene para cada uno de los caracteres del teclado (el orden alfabético). De esta forma, la letra «B» se considera menor que la «C» etc...
En las comparaciones con variables de tipo numérico, el ordenador tiene en cuenta en primer lugar los signos y luego los valores. En el caso de que los elementos a comparar sean de diferente signo, resulta siempre mayor el positivo. Si los elementos son del mismo signo, se analiza el valor absoluto (valor sin signo) de las variables. Si las cantidades son positivas, se toma como mayor la de valor absoluto superior, y si son negativas, la de valor absoluto inferior.
En el caso de las cadenas de caracteres, esta operación de comparación se realiza de izquierda a derecha, con el convenio de que si dos cadenas son idénticas en lo que se refiere a sus primeros caracteres, es considerada como menor la más corta. De esta forma, la cadena "ABC" se considera siempre menor que la "ABCD".
El juego de caracteres completo del Spectrum puede encontrarse al final de la Obra, aunque normalmente debe bastarnos con saber que los números son para el Spectrum menores que todas las letras, y que las letras mayúsculas son menores que las minúsculas.

LAS DECISIONES SE COMPLICAN

Hasta ahora hemos hablado de las comparaciones más simples, las de tipo unitario, en las que manipulamos un solo par de elementos. Manteniendo la misma estructura, podemos escribir instrucciones mucho más potentes apoyándonos en los OPERADORES RELACIONALES, también llamados OPERADORES LOGICOS.
Estos operadores, establecen la relación que debe cumplirse entre los diferentes elementos



En las comparaciones realizadas con datos numéricos, primero se tiene en cuenta el signo y en caso de ser éstos iguales, su valor absoluto.

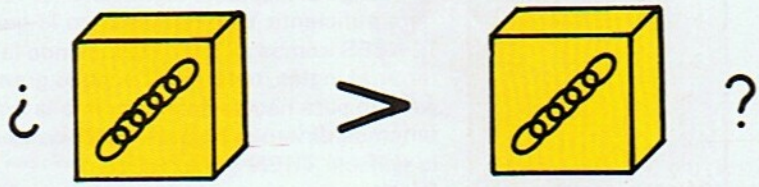
que componen la instrucción de comparación, siguiendo el formato:

IF condición OPERADOR condición ... **THEN** instrucción

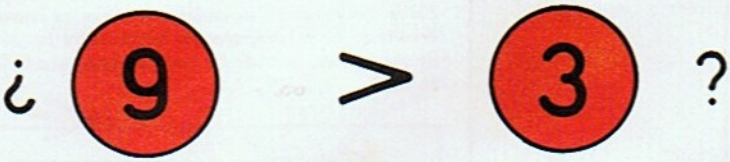
En este formato mantenemos una estructura parecida a la de los **IF** simples, pero haciendo intervenir diferentes condiciones relacionadas con los correspondientes operadores. Los tres tipos de operadores lógicos son: **AND, OR** y **NOT;** su traducción al castellano es **Y, O** y **NO,** respectivamente; su significado lo estudiaremos a continuación mediante unos ejemplos.
Comenzaremos hablando del operador **AND.** Su-

No debemos establecer comparaciones entre elementos de distinta clase (números y cadenas).

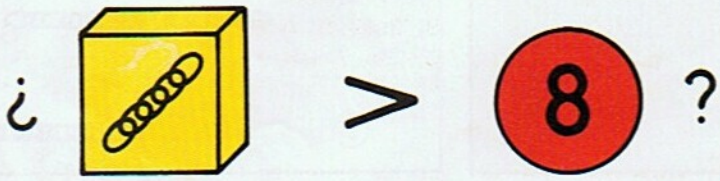
CORRECTO



CORRECTO



INCORRECTO



Si comparamos dos cadenas que comienzan por la misma secuencia de caracteres, se considera menor la más corta.

*

En la comparación entre variables de cadena, se sigue el orden establecido de códigos de caracteres del Spectrum, que podemos encontrar detallado al final de nuestra Obra o en el manual del aparato.

*

Al comparar variables de tipo numérico, se tiene en cuenta primero su signo y después su valor absoluto.

*

Los operadores lógicos permiten establecer más de una situación a evaluar, para determinar un salto en la secuencia del programa, o la ejecución de determinadas instrucciones.

*

Los posibles operadores de comparación son: =, <, >, <=, >= y <>.

El operador AND *necesita que los dos sucesos que relaciona sean verdaderos para que la condición se cumpla.*

pongamos que tenemos que decidir si vamos o no al cine, dependiendo el hecho de dos condiciones:

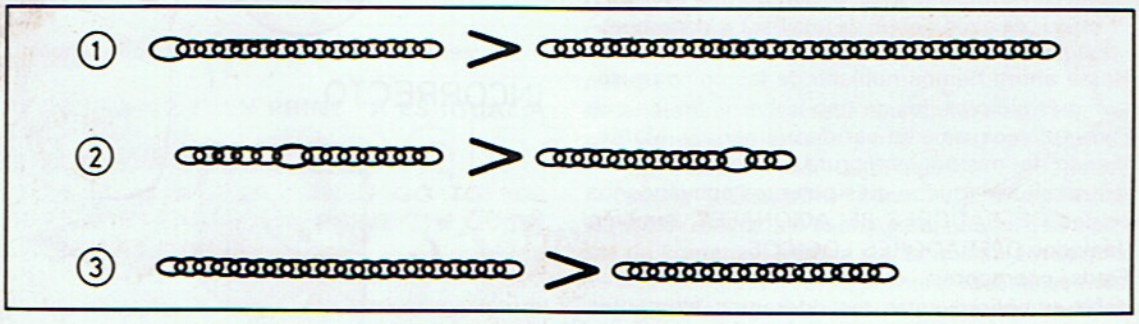
1. TENER DINERO PARA LAS ENTRADAS
2. NO HABER VISTO LA PELICULA

Evidentemente, tenemos que relacionar ambas condiciones con el operador **AND** (Y), ya que las dos deben ser ciertas al mismo tiempo, es decir, no vamos a ir si no tenemos dinero para las entradas ni, por supuesto, si la película la hemos visto ya. Por tanto, la decisión por parte del ordenador se tomará por medio de una instrucción que tenga el siguiente significado: SI tenemos dinero suficiente Y no hemos visto la película ENTONCES iremos al cine. Traduciendo la frase anterior al inglés, tendremos andado gran parte del camino para hacérselo entender a la máquina: **IF** tenemos dinero suficiente **AND** no hemos visto la película **THEN** iremos al cine.
Para terminar de trasladar nuestro problema a un programa BASIC, sólo nos resta hacer comprender al ordenador las condiciones "tenemos dinero suficiente" y "no hemos visto la película". ¡Manos a la obra! Supongamos que **D** es el dinero que tenemos, y **P$** una película que queremos ver. Vamos a escribir el siguiente programa:

```
  10 REM IR O NO IR, HE AQUI EL DI
LEMA
  20 CLS
  30 INPUT 'Dinero:';D
  40 PRINT 'Tenemos: ';D;' Ptas.'
  50 INPUT 'Pelicula:';P$
  60 PRINT 'y queremos ir a ver: '
;P$'
  70 IF D>=350 AND P$<>'TRON' THEN
  GO TO 120
  80 PRINT 'NO PODEMOS IR porque '
;
  90 IF D<350 THEN  PRINT 'NO tene
mos DINERO ';
 100 IF P$='TRON' THEN  PRINT 'YA
la hemos VISTO';
 110 GO TO 130
 120 PRINT 'PODEMOS IR SIN PROBLEM
AS.'
 130 PRINT ''''PULSA UNA TECLA'
 140 PAUSE 0
 150 GO TO 20
```

En la comparación de cadenas se tiene en cuenta el resultado de la comparación lógica entre sus elementos y, de ser iguales, cual de las cadenas comparadas es más larga.

De la ejecución de este programa podemos sacar algunas conclusiones interesantes. Ante todo diremos que como datos fijos (constantes) hemos



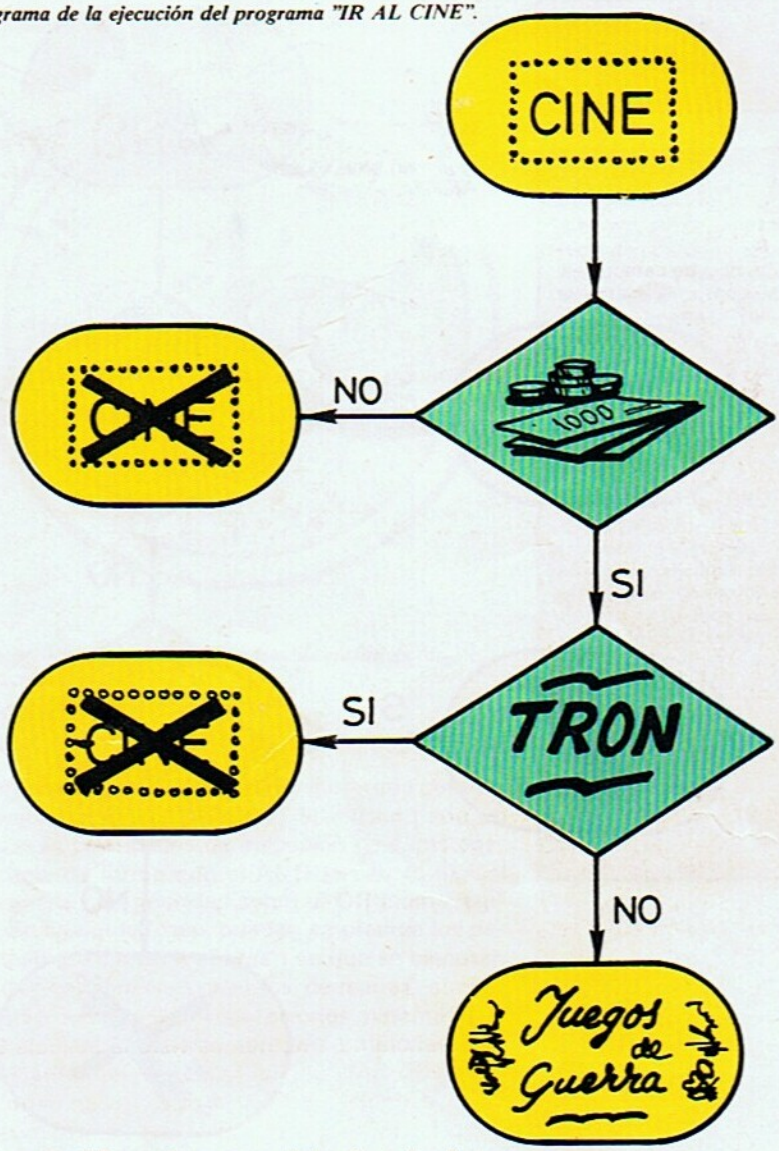
introducido el precio de la localidad y una de las películas que ya hemos visto (la única que el programa va a considerar).
Lo deseable para que el programa fuera un práctico consejero a la hora de resolvernos la tarde del fin de semana, sería poder contar con un archivo en la memoria de todas las películas que hemos visto. En el caso de que seamos muy aficionados al séptimo arte, esto podría traernos problemas de falta de memoria; por otro lado, nuestra única intención es que el programa nos sirva de ejemplo de la forma en que debemos codificar la toma de decisiones para el Spectrum. Como ya sabemos, las líneas 10 y 20 se ocupan, respectivamente, de dar información sobre el contenido del programa y de borrar la pantalla. Las líneas 30 a 60 efectúan la entrada de datos del programa y representan la información en la pantalla. Las variables que utilizamos para esta entrada son: **D** para el dinero de que disponemos y **P$** para el título de la película candidata. En la línea 70 establecemos la más ardua de las decisiones:

```
70 IF D>=350 AND P$<>"TRON" THEN
GO TO 120
```

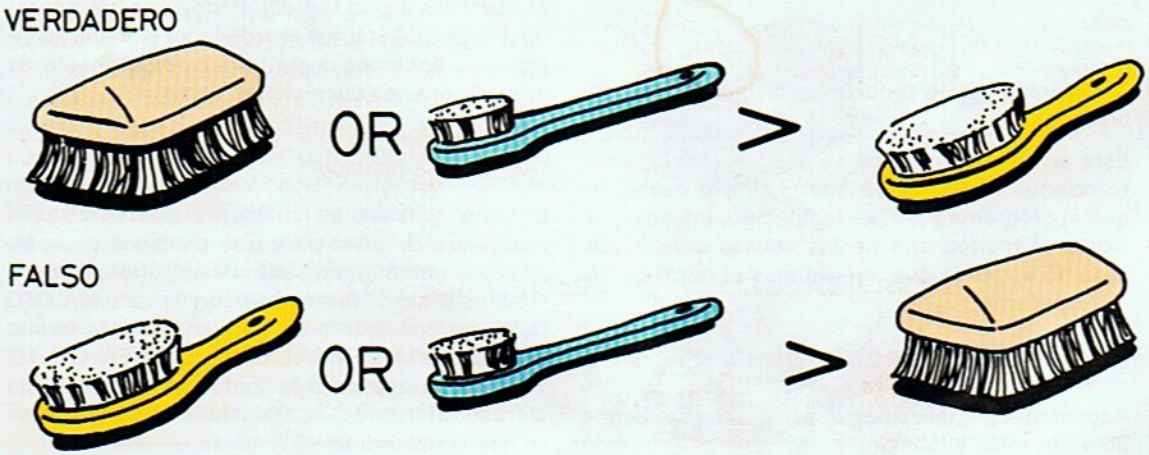
De acuerdo con dos condiciones vinculadas por el operador lógico **AND**, determinamos definitivamente en esta línea si podremos ir o no al cine. Si ambas condiciones resultan ser ciertas (no basta con que lo sea una sola), en base a los datos que introducimos (**D** y **P$**), el programa sigue a partir de la línea 120, señalándonos que no existe ningún problema. En caso contrario, el programa continuará en secuencia, haciendo caso omiso de la sentencia **GO TO**.
Una vez decidido que no podemos ir al cine, entramos en el análisis de los motivos. No existen más que tres posibilidades:

1. No tenemos dinero.
2. Ya hemos visto la película.

Diagrama de la ejecución del programa "IR AL CINE".



El operador **OR** *necesita que uno de los dos (o los dos) sucesos que relaciona se cumplan, para que la condición se cumpla.*



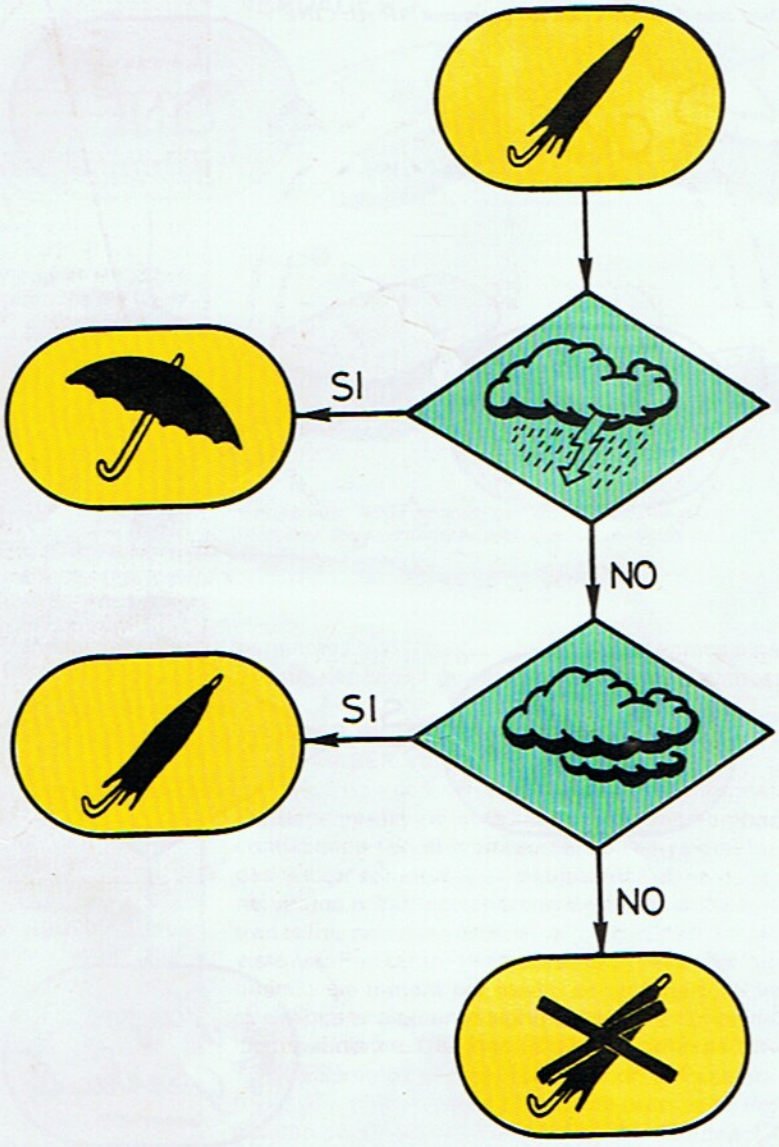


Diagrama de la ejecución del programa "CONDICIONES METEOROLOGICAS".

3. Además de no tener dinero, hemos visto la película.

Este análisis de motivos se concreta en las instrucciones 90 y 100, ya que el simple hecho de que el programa pase por la línea 80, implica que existe al menos una de las causas suficientes para no ir, por lo que imprimimos el mensaje de desesperanza:

80 PRINT "NO PODEMOS IR porque ";

Acto seguido analizamos el porqué hemos llegado a tan triste situación:

90 IF D<350 THEN PRINT "NO tenemos DINERO ";
100 IF P$="TRON" THEN PRINT "YA la hemos VISTO";

Si tenemos menos de 350 pesetas, se imprimirá el mensaje "NO tenemos DINERO", y si hemos indicado como candidata la película "TRON", lo hará el mensaje de "YA la hemos VISTO". Puede también darse la posibilidad de que concurran ambas circunstancias, en cuyo caso se imprimirán los dos mensajes.
Entramos por último en el análisis de las líneas 130 a 150, que nos deparan una sorpresa: la aparición en escena por primera vez de la palabra clave **PAUSE.**

130 PRINT"'"PULSA UNA TECLA"
140 PAUSE 0
150 GO TO 20

La sentencia **PAUSE** se emplea para detener momentáneamente la ejecución del programa, por un espacio de tiempo determinado: el especificado a continuación de la palabra clave. Dicha pausa se expresa en cincuentavos de segundo, de forma que si deseamos una detención de un segundo deberemos indicar **PAUSE 50,** si dos segundos **PAUSE 100,** etc...
En cualquier caso, el programa continuará al pulsar una tecla, a pesar de no haber concluido su tiempo completo de espera. Otra posibilidad es la de provocar una parada en el programa por tiempo indefinido, de la cual la única forma de salir, para que la ejecución continúe, es pulsando una tecla. Esta última opción es la que lleva un cero como parámetro de la sentencia **PAUSE (PAUSE 0).**
En nuestro ejemplo, nos servimos de esta instrucción para provocar una pausa indefinida, en la que se muestran los resultados impresos y se solicita que pulsemos una tecla para volver a ejecutar el programa.
De forma similar a la que hemos actuado en el programa propuesto anteriormente, podemos diseñar otro muy parecido en el que emplearemos el operador lógico **OR** (O). Podría ser un programa en que pidiéramos al ordenador que nos aconsejara si debemos coger la gabardina al salir de casa. Ahora, las condiciones serán:

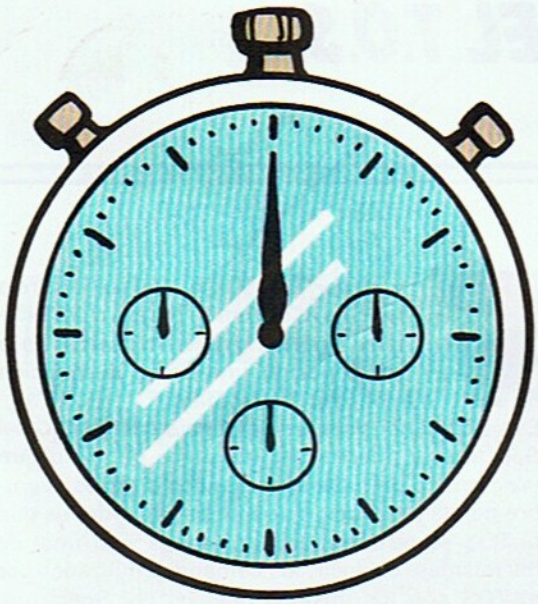
1. Si llueve.
2. Si está nublado.

En esta ocasión, es suficiente que se cumpla cualquiera de ellas para que debamos proveernos de la indumentaria apropiada, correspondiéndose este caso claramente con la relación **OR.**
Debemos codificar la instrucción 70 de la forma: **70 IF LLUVIA OR MAL TIEMPO THEN GO TO 120.** Esta línea sería la traducción al BASIC de la frase «SI llueve O está nublado ENTONCES cogemos la gabardina».

Para facilitar la codificación de la instrucción hemos escogido dos variables cuyo nombre está directamente relacionado con el valor que contienen: LLUVIA y MAL TIEMPO; estas variables son numéricas, de forma que para la entrada de datos en los **INPUT**, debemos adoptar el convenio de considerar como «1» si el fenómeno atmosférico se produce y como «0» si no. Una novedad en la instrucción 70 es la de que no aparece el símbolo de igualdad detrás de las variables:

LLUVIA=1 y MAL TIEMPO=1

Esto podemos hacerlo gracias a que el ordenador supone que, al no existir el símbolo de igualdad, es nuestro deseo tratar estas variables numéricas como variables lógicas. Las variables lógicas adoptan para la comparación dos únicos valores: VERDADERO (diferente de 0) y FALSO (0).
Al escribir la instrucción de esta forma, conseguimos que el programa bifurque a la línea 120 cuando LLUVIA o MAL TIEMPO sean, las dos o cualquiera de ellas, diferentes de cero, es decir, verdaderas en sentido lógico.
La codificación del resto del programa será muy simple, puesto que sólo deberemos realizar modificaciones en las sentencias **PRINT** del programa, según las condiciones meteorológicas que se hayan producido, de manera similar a como procedimos con nuestro programa del cinéfilo.
Hablaremos por último del operador lógico **NOT**, que es el complementario (negación lógica) de los operadores lógicos que hemos visto hasta ahora. La verdadera utilidad del operador **NOT** es la de clarificar nuestras intenciones respecto a las condiciones que se avalúan en un programa. Expliquemos esto más claramente: supongamos que determinado suceso depende de una sola condición; por ejemplo, que ir a trabajar depende exclusivamente de que no sea domingo. La sentencia **IF** que deberíamos utilizar para codificar este suceso sería: SI hoy es diferente de domingo ENTONCES vamos a trabajar.
Para traducir esta instrucción al lenguaje BASIC, emplearemos una variable lógica que denominaremos DOMINGO, cuyo valor será cero en caso de que no sea domingo, y uno si hoy es domingo. La instrucción de decisión se podría concretar de la siguiente manera: **IF DOMINGO <> 1 THEN vamos a trabajar** o **IF DOMINGO = 0 THEN vamos a trabajar.** En este caso hubiera quedado mucho más clara una decisión del tipo. «Si hoy NO es domingo ENTONCES vamos a trabajar»; la traducción directa de esta frase al BASIC sería: **IF NOT DOMINGO THEN vamos a trabajar.** De lo dicho, sin duda habremos sacado la siguiente conclusión: **NOT** variable, tiene exactamente el mismo efecto que: variable **<> 0.** Para terminar, diremos que se pueden construir complicadas estructuras de comprobación de condiciones, utilizando combinaciones de los operadores lógicos. Del mismo modo que para las operaciones matemáticas, el Spectrum tiene un orden de prioridades de ejecución de estos operadores, de forma que el **AND** (producto lógico) se ejecuta con prioridad sobre el **OR** (suma lógica). En cualquier caso, pueden emplearse los paréntesis para alterar el orden en que se ejecutan normalmente las operaciones, de manera similar a lo que ocurre con las operaciones matemáticas. De todas formas, sobre este tema ahondaremos próximamente.



La sentencia PAUSE *establece una detención en la ejecución del programa.*

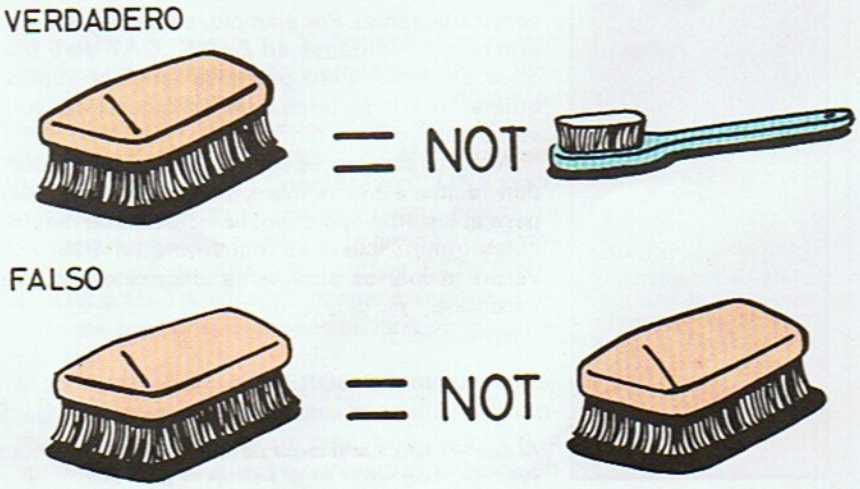
¡!

Con **PAUSE 0**, se provoca una parada indefinida, de la que sólo se puede salir cuando el usuario pulsa una tecla.

*

La instrucción **PAUSE**, produce una interrupción en la ejecución del programa, por el espacio de tiempo especificado como argumento de la sentencia, expresado en cincuentavos de segundo.

El operador NOT *es equivalente en su efecto a la comparación "distinto que".*



EL T.O.S.

EL disco INVESDISK 200 se controla gracias al sistema operativo TOS (Timex Operating System); veremos a continuación una breve descripción de sus comandos fundamentales.

Estos comandos son una extensión del BASIC del Spectrum y pueden ser utilizados dentro de un programa o bien como comandos directos, siempre con la precaución de marcar las órdenes que al TOS se refieran, con un asterisco al final de las mismas. Es decir, los comandos utilizados por el TOS son los mismos que los del Spectrum, pero marcados con el asterisco (aunque algunos no realizarán la misma función).

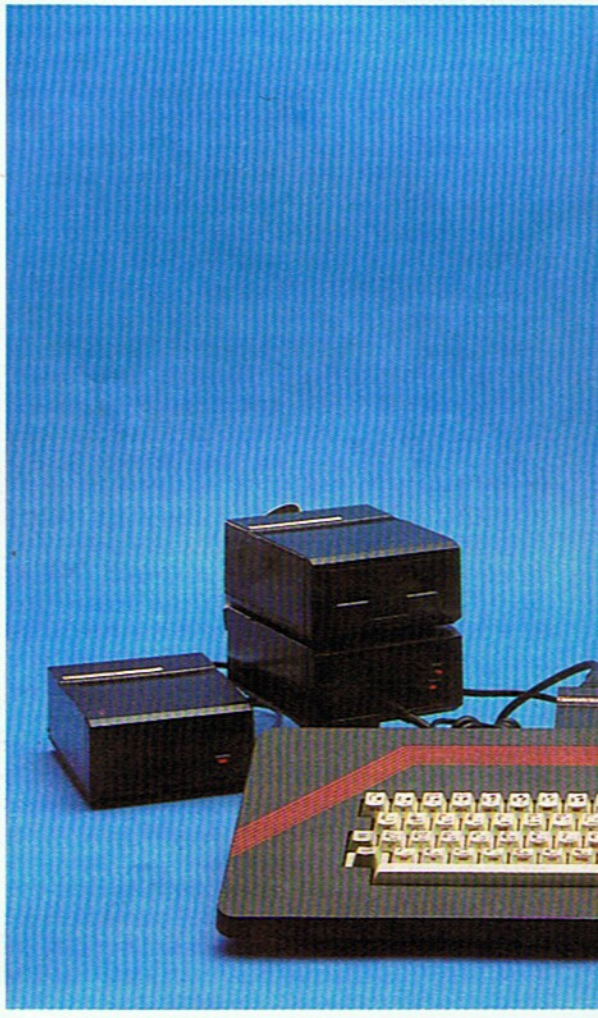
NOMBRES DE PROGRAMAS Y FICHEROS

Podemos utilizar cualquier nombre que deseemos con las siguientes limitaciones:

* No tendrán una longitud mayor que ocho caracteres; opcionalmente, a continuación del nombre y separados por un punto (.) se pueden incluir otros tres caracteres, que nos sirven para diferenciar tipos de programas; estos apéndices de los nombres se denominan extensiones, (**ext** abreviadamente). Por ejemplo, se utiliza **.BAS** en programas realizados en BASIC **.DAT** para bloques de datos. Pero podemos emplear cualesquiera tres caracteres que consideremos convenientes.

* Algunos caracteres como ↑, +, . o ?, no se pueden incluir en el nombre, por estar reservados para el sistema operativo; la inclusión de mayúsculas o minúsculas es indiferente (el TOS convertirá todos los caracteres alfabéticos en mayúsculas).

Cuando almacenamos información mediante un casete, no hay manera rápida de saber lo que tenemos grabado, ni lo que esto ocupa en una determinada cinta, a no ser que lo hayamos anotado cuidadosamente. En la unidad de discos, este problema se resuelve fácilmente con el uso de **CAT ***. Este comando nos va a informar, mediante una impresión en la pantalla, de todos los programas que se encuentran en la cara del disco con la que estamos trabajando. Proporciona además los siguientes datos: la longitud de cada uno de ellos, el número de Kbytes ocupados redondeados al entero superior más próximo, si están protegidos o no, el número de Kbytes que todavía quedan libres y los utilizados y, cuando se trata de un fichero, si está abierto o no.



El disco es sin duda la mejor de las soluciones para convertir el Spectrum en un sistema de gestión.

BITS

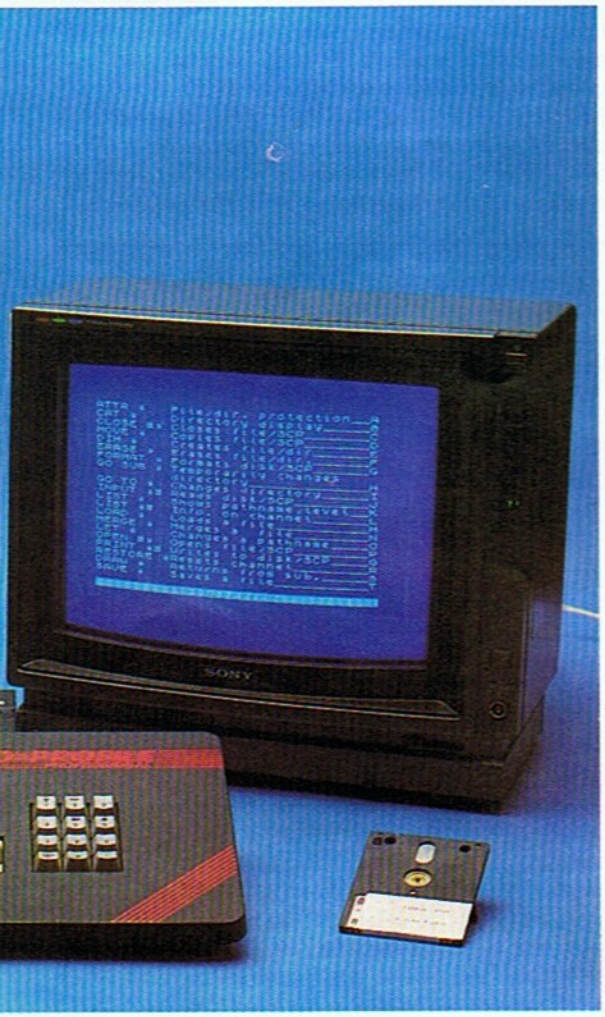
En algunas ocasiones, por ejemplo durante la ejecución de un **INPUT**, hacer desaparecer el cursor puede mejorar la presentación de nuestro programa, o incluso despistar a un posible «pirata»; para ello basta con incluir la siguiente instrucción: **POKE 23617,226**

*

Si alguno todavía duda de los sobresaltos que puede provocarnos el Spectrum, que teclee: **POKE 23659,1**. ¡Tranquilo!... pulsa **ENTER**.

El sistema operativo puede organizar los catálogos en forma jerárquica, es decir, como si se tratara de un árbol invertido, con la raíz en la parte superior y las ramas hacia abajo, Si al ajecutar el comando **CAT *** encontramos ficheros con la extensión **.DIR,** esto indica que tenemos un directorio dentro de otro en alguna de las distintas ramas del árbol. El TOS permite la creación de dieciséis ficheros de direcciones en cada cara del disco y de ocho niveles distintos en los que anidar los archivos. Se puede acceder a las ramas del árbol mediante los comandos **GOSUB *** o **GO TO * "nombre.DIR".**

Para la carga desde el disco se utiliza la sentencia **LOAD,** cuya sintaxis es: **LOAD * "nombre"** o, si hemos hecho uso de los tres caracteres opcionales de extensión, **LOAD * "nombre.ext".** Su empleo es muy similar a la sentencia homónima para el casete. Es importante tener en cuenta que con el disco no podremos cargar ficheros sin indicar el nombre, tal como haríamos con el casete; por tanto, el TOS no identificará la orden **LOAD * ""**. Siempre hará falta especificar el nombre del programa. Esto no produce ningún contratiempo pues, aunque no recordemos el nombre exacto, va a ser fácil averiguarlo utilizando el comando **CAT *.**



SENTENCIAS INCORPORADAS POR EL TOS

ATTR *	INPUT *
CAT *	LIST *
CLOSE # *	LOAD *
MOVE *	MERGE *
DIM *	LET *
ERASE *	OPEN # *
FORMAT *	PRINT *
GOSUB *	DRAW *
GOTO *	SAVE *

SAVE * es el comando inverso a **LOAD *.** Hay que destacar que cuando grabemos un programa o un fichero al que queremos dar un nombre que ya figura en el directorio del disco, el TOS nos advertirá de esta circunstancia mediante el mensaje:

"nombre del programa" **already exists**
Superse de (Y/N)?

Si nuestra respuesta es **Y,** borrará el programa existente ya mencionado y grabará la nueva versión desde la memoria. Si por el contrario pulsamos **N** ignorará la operación **SAVE *.**

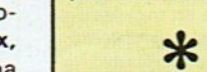
De forma similar al casete o al ZX Microdrive podemos utilizar **SAVE * "nombre.ext" LINE xxxx,** que tiene el efecto de grabar nuestro programa con autoejecución a partir de la línea **xxxx.** También podremos utilizar los sufijos **SCREE N$, CODE** o **DATA** para almacenar pantallas, bloques de memoria o dimensionados, respectivamente.

El funcionamiento de **MERGE *** es también análogo al correspondiente comando del BASIC Sinclair estándar: mezcla un programa nuevo y sus variables, con el que se encuentre presente en

¡!

Recordemos que los dos tipos fundamentales de ficheros se denominan SECUENCIALES y ALEATORIOS. Estos se diferencian por el modo de acceso a cada uno de sus campos.

Para el acceso al cualquier campo de un fichero secuencial, es necesario leer previamente todos los campos anteriores a él.



El acceso a un fichero aleatorio se realiza libremente a cualquiera de sus campos, designándolo por su número de orden dentro del fichero, de forma similar a los dimensionados BASIC.

BITS

El espacio total de memoria disponible, incluyendo el dedicado a gráficos definidos por el usuario, es fácil de calcular mediante una sencilla llamada a la ROM del sistema. En cualquier momento podemos ejecutar la siguiente instrucción: **PRINT 65536-USR 7962;'' Bytes libres''**. Si el Spectrum es de 16 K, sustituiremos el 65536 por 32768.

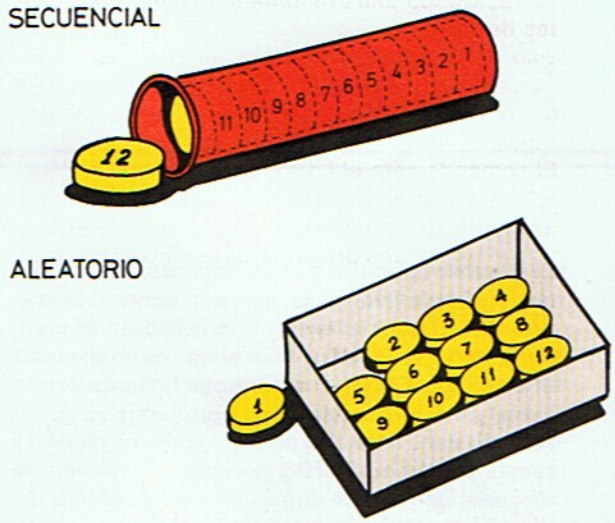
*

Incluir sonidos dentro de nuestros programas los hará sin duda mas entretenidos. A continuación se listan tres efectos sonoros que podremos aplicar muy fácilmente.
Caída: **10 FOR N=0 TO 60 STEP .5: BEEP .03,N: NEXT N**
Sirena: **10 BEEP .35,24: BEEP .35,12: GOTO 10**
«Click-Clack»: **10 FOR N=0 TO 200: BEEP .01,20: PAUSE 10: NEXT N**

*

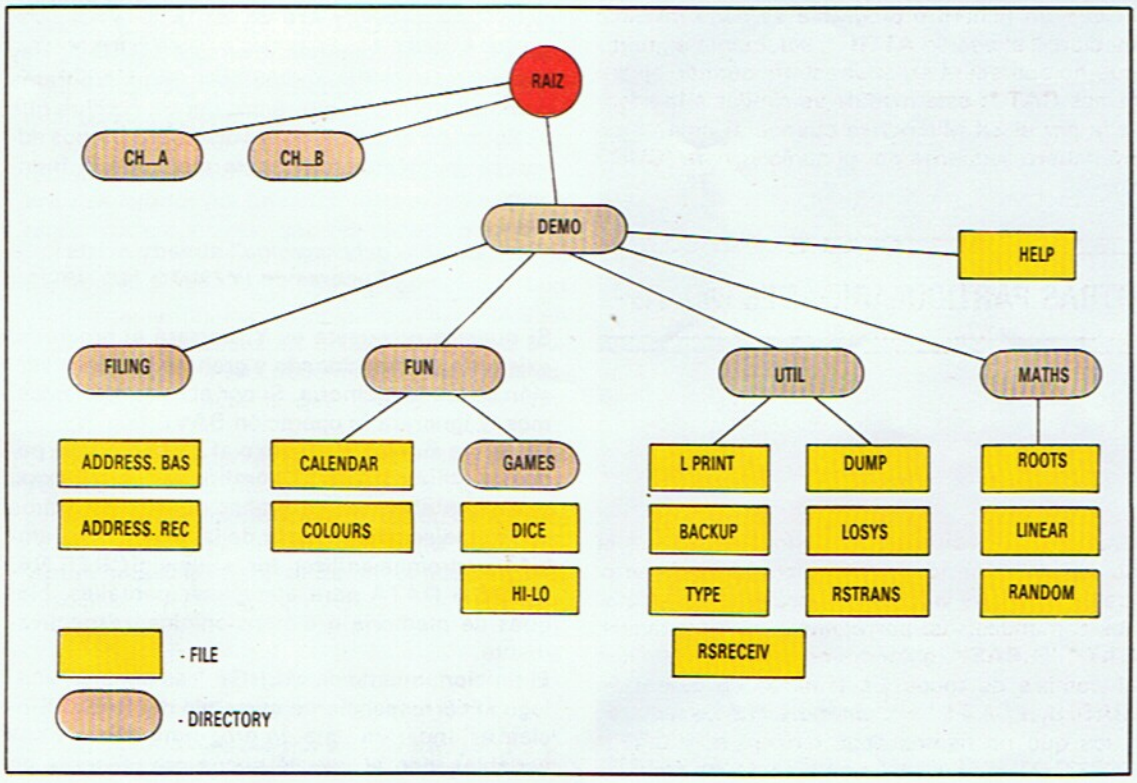
Si deseamos asegurarnos desde dentro de un programa, que una toma de datos es realizada en mayúsculas, nuestro problema se reduce a ejecutar la siguiente instrucción: **POKE 23658,8**. Si por el contrario lo que queremos es conectar el modo minúsculas, sólo tenemos que cambiar el valor del **POKE** a cero.

la memoria del Spectrum. Lógicamente, no es posible realizar un **MERGE** con un bloque de bytes o datos, sino sólo con programas. La sintaxis de la instrucción es: **MERGE * ''nombre.ext''**.
El comando **FORMAT** es probablemente el más importante de todos, puesto que es obligatoria su ejecución previamente a la utilización de cualquier disquete virgen.
Esta es la única sentencia del TOS en la que se debe especificar el nombre de la unidad de disco con la que vamos a trabajar. Para la identificación del *drive* (unidad de disco) afectada, utilizaremos las primeras letras del abecedario, asignándosele la **A** a la primera unidad, **B** a la segunda, **C** a la tercera y **D** a la cuarta. La sintaxis completa de la instrucción es: **FORMAT * ''nombre de la unidad '' TO ''nombre del disquete''**. Esta sentencia se introduce por el teclado como comando directo.
Cuando se ejecuta el comando **FORMAT ***, el sistema operativo divide la superficie del disquete insertado en la unidad de disco en cuarenta pistas de cuatro Kbytes, y cada pista en sectores. Es decir, lo prepara para que la información que allí va a ser almacenada, se distribuya ordenadamente y luego pueda tener fácil acceso a ella. El «formateado» de un disco dura poco más de treinta segundos, y el sistema operativo que se copia en el disquete ocupa 16 K. Otras 4 K se utilizan para el directorio o catálogo del mismo; de ahí que de las teóricas 160 K de cada cara del disquete, sólo podamos aprovechar 140 K.
Es importante tener cuidado con este comando, pues al ejecutarlo se destruye totalmente el contenido del disquete. De todos modos, el TOS se cerciora de que queremos «formatear» el disque-



El acceso a las pastillas de un tubo es secuencial; por el contrario, de hallarse dispuestas en una caja, podríamos hablar de acceso aleatorio.

El disquete de demostración que se suministra con la unidad de disco, utiliza varios directorios dispuestos en un sistema de árbol.



te, realizando una pregunta para confirmarlo antes de ejecutar el comando. Así mismo, cuando sólo disponemos de un *drive,* el TOS nos irá informando de cuando debemos retirar el disquete a «formatear» para insertar el del sistema operativo.

El comando para el borrado de ficheros es **ERASE *** y su sintaxis correcta es: **ERASE * ''nombre.ext''**. Antes de ejecutarlo, el sistema operativo realizará una pregunta de confirmación de la forma:

ERASE ''nombre.ext'' Y/N?

Para que la sentencia **ERASE *** tenga efecto es necesario que el fichero que queremos eliminar no se encuentre protegido. Sabremos si lo está ejecutando **CAT *** y mirando en la columna marcada con **P**; si encontramos una letra **P** en la línea correspondiente al fichero a borrar, esto será indicativo de que éste se encuentra protegido. Uno de los comandos del TOS que en BASIC Sinclair estándar tienen un significado absolutamente diferente es **ATTR ***. Este comando permite proteger, desproteger, hacer invisible o visibilizar un determinado fichero del directorio. Su sintaxis es:

ATTR * ''nombre.ext''p - protege el fichero.
ATTR * ''nombre.ext''i - hace invisible el fichero.
ATTR * ''nombre.ext''u - desprotege el fichero.
ATTR * ''nombre.ext''v - visibiliza el fichero.

El que un fichero o programa se haga invisible mediante el uso de **ATTR ***, solamente significa que no aparecerá en el directorio cuando ejecutemos **CAT ***; esta medida es similar a la adoptada por el ZX Microdrive cuando el nombre de un fichero comienza por el carácter cero (**CHR$ 0**).

OTRAS PARTICULARIDADES DEL TOS

Una de las posibilidades que ofrece el TOS es la de incluir en el nombre del programa o fichero, los caracteres + o ? cuando usamos determinados comandos. Así por ejemplo, si ejecutamos: **CAT * ''+.BAS''**, obtendremos un directorio con el nombre de todos los ficheros de extensión **.BAS**; con **CAT * ''+''**, obtendremos los ficheros a los que no hemos dado extensión; y **CAT * ''R???????.+''**, buscará una lista de los archivos o programas que comienzan por R. Estos caracteres se pueden utilizar con los comandos **ERASE *, ATTR *, MOVE *, LET *** y **CAT ***.

Mediante la instrucción **LET * ''nombre antiguo.ext'' TO ''nuevo nombre.ext''**, conseguiremos cambiar el nombre de un fichero por otro nuevo. Continuando con las alteraciones de denominación, **GOSUB * ''nombre de la unidad''d** se utiliza para cambiar la unidad de disco a que se hace referencia, donde el nombre de la unidad se designa con las letras A, B, C o D.

Las sentencias básicas utilizadas por el TOS para el control del disco, son muy similares a las empleadas por el ZX INTERFACE 1, para la gestión del Microdrive.

El comando **CAT *** *proporciona una amplia información sobre el contenido de un disquete.*

```
DEMO
Level 0   Drive A:

Name      Typ      Size  Alloc  S P
-----------------------------------
ALFA      DIR     41770    50K    P
COCHES              40     1K
CHUZOR             100     1K
FCHUZOR            060     1K    P
AUTOS     ENT      142     1K    P
AUTOS     LEC      106     1K    P
DIRECC    DAT    12000    14K
2+2             2600     3K
DIRECC    DEM      959     1K
CHULIN    DIR     4094     5K
INTERF    BAS   101118    10K    P
IDIOA     CAR       28     1K    P
AFON              037     1K
79585     DIR       35     5K

Max 140K   Cur 93K   Rem 47K

0 OK, 0 1
```

BITS

Este sencillo programa proporcionará dibujos lineales en la pantalla, siempre diferentes y sin el fantasma del desagradable mensaje *Integer out of range.*

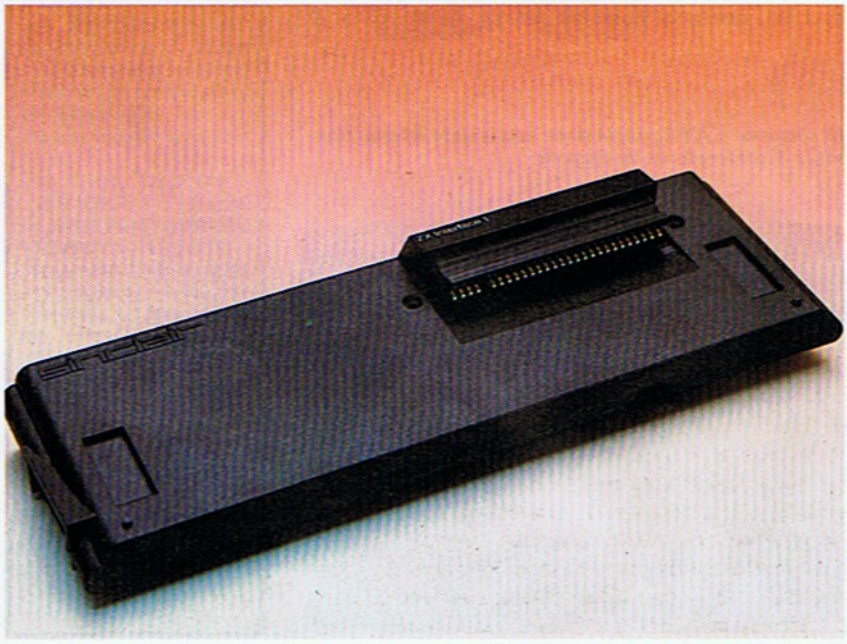
```
10 LET X=RND*255: LET Y=RND*175: INK RND*7: PLOT X,Y
20 LET A=(RND*255)-X: LET B=(RND*175)- Y: DRAW A,0: DRAW 0,B
30 GOTO 10
```

Prueba ahora eliminando la línea 20.

Una vez realizada la operación de «formateado», el disquete tiene 140 Kbytes netos para el usuario por cada cara.

Para finalizar con este breve repaso al TOS, mencionaremos que la copia de programas se consigue mediante **MOVE * "nombre1.ext" TO "nombre2.ext"**. La potencia del sistema operativo del INVESDISK es aún mayor que lo aquí expuesto. Se dispone de un total de dieciocho comandos destinados al disco, gracias a los cuales éste se convierte en un periférico muy útil y eficaz. Por último cabe añadir que el apoyo que presta el TOS al programador, va desde las preguntas de confirmación ante operaciones tan delicadas como el **FORMAT** o el borrado, hasta la emisión de un total de cuarenta y seis tipos diferentes de mensajes de error, todos ellos detallados en el manual de operación que acompaña a la unidad de disco.

El sistema INVESDISK 200 es incompatible con el ZX INTERFACE 1, y por tanto no permite la utilización simultánea de Microdrives.



FICHEROS DE DATOS

La principal ventaja de utilizar la unidad de discos frente a otros sistemas de almacenamiento, es la posibilidad de manejar ficheros de acceso aleatorio (en el ZX Microdrive, como con algunos sistemas de cinta, sólo se pueden manejar ficheros sencuenciales).

Cada uno de los componentes de un fichero se denomina *RECORD* o CAMPO (en ocasiones REGISTRO); la forma de acceder a cada uno de los campos de un fichero, es la que marca el tipo de éste. Fundamentalmente, existen dos tipos de ficheros: SECUENCIALES y ALEATORIOS.

Los primeros se caracterizan porque para acceder a cualquiera de sus campos es necesario haber leído previamente todos los anteriores. Imaginemos un tubo de pastillas dulces; si aquella cuyo sabor estamos buscando se encuentra al final del tubo, nos veremos obligados a sacar todas las pastillas anteriores para llegar hasta la que buscamos. Este es el tipo de acceso secuencial, utilizado en el ZX Microdrive.

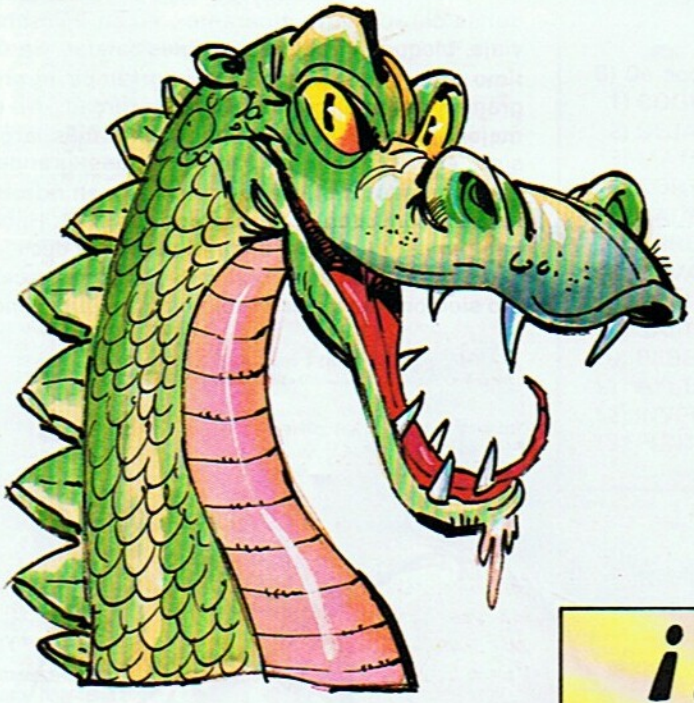
Cuando podemos acceder a cada uno de los *records* de un fichero sin pasar por los anteriores, decimos que este fichero es de acceso aleatorio. Siguiendo el ejemplo anterior, podríamos imaginar que las pastillas han sido distribuidas en una caja formando un solo piso, gracias a ello podremos llegar directamente a cada una de ellas sin necesidad de desplazar el resto. Este tipo de ficheros es sin duda el más conveniente puesto que, aunque más lentamente, simula el tipo de acceso que en el BASIC se tiene a los dimensionados.

Para el control de ambos tipos de ficheros, el TOS emplea las sentencias **OPEN # *, CLOSE # *, INPUT *** y **PRINT ***. Cuando se trata de ficheros aleatorios, el *record* a acceder dentro del fichero se señala a continuacion del sufijo **AT.**

AVENTURA INTERMINABLE

ENTRE los programas que gozan de mayor aceptación dentro de los destinados a microordenadores, se cuentan los del tipo AVENTURA. Gracias a ellos nos trasladamos a un mundo irreal, en el que el destino de un personaje y el final de una historia depende directamente de nosotros, de nuestras decisiones y comportamiento. Este argumento, muy similar al de la conocida película de fantasía «La Historia Interminable», tiene también brillantes exponentes en el software destinado a nuestro aparato. Tal es el caso de programas tan conocidos como «El Hobbit» **(Melbourne House)** u otros muchos. Sin embargo, no debemos pensar que la realización de este tipo de programas nos está vedada a los desconocedores del código máquina. También en BASIC se pueden conseguir algunas «aventuras» interesantes.

Sirva como botón de muestra el programa que figura a continuación; aunque hemos de tener en cuenta que se trata de una versión reducida, puesto que de no ser así, la gran ocupación de espacio haría imposible su publicación.



LA HISTORIA

Este juego-aventura nos permitirá explorar nuevos y extraños parajes, encontrar tesoros, luchar contra monstruos; tendremos que sobrevivir en un inhóspito y engañoso desierto. En pocas palabras: si queremos acabar con vida, hemos de resolver todos y cada uno de los problemas que se nos plantearán en el nuevo mundo al que vamos a entrar. Obviamente, nosotros seremos los protagonistas de esta «historia interminable».

El Spectrum nos describirá los lugares que vayamos visitando y los objetos que aparezcan. Comenzaremos en un determinado lugar del condado de Kerkovia, perteneciente al territorio de nuestro gran enemigo: ASHAMM, el dragón verde.

Nuestro objetivo es hallar los tesoros de la princesa Shirka y depositarlos en el punto de partida. Para ello tendremos que hacer uso adecuado, en el momento oportuno, de los objetos que encontremos a nuestro paso, y con ellos salir airosos de los múltiples problemas con los que nos vamos a topar. Pero... ¡CUIDADO!, los sangrientos piratas del país vecino están al acecho.

Hemos de tener en cuenta, que todos los objetos que nos encontramos, por extraños que nos parezcan, tienen una utilidad y puede que nos sean necesarios en algún momento de la aventura. Veremos un ejemplo a modo de pista: si intentamos atravesar el lago del oasis del desierto, sin proveernos de un tubo hueco que poder utilizar para respirar mientras andamos por el fondo, pereceremos en el intento.

Lamentablemente, no podremos portar todos los objetos que nos vayamos encontrando, por lo que deberemos seleccionar con cuidado nuestro equipaje. Ante todo hemos de utilizar la imaginación, tengamos en cuenta que somos un personaje de fantasía que ¡hasta puede comer lámparas! Sin embargo, frenemos en lo posible nuestra hambre pantagruélica, puesto que esa misma lámpara nos puede servir para ir hacia la parte norte de la casa en que comienza nuestra aventura, donde reina la oscuridad; al intentar entrar allí, siem-

¡!

Las subrutinas han sido situadas al comienzo del programa, para conseguir un acceso más rápido a las mismas.

*

Los programas del tipo AVENTURA no han de estar necesariamente escritos en código máquina; también en BASIC podemos conseguir resultados bastantes aceptables.

*

El factor aleatorio influye en el programa, gracias a lo cual cada nueva aventura será diferente a las anteriores.

pre que no vayamos provistos de la lámpara, el programa emitirá el mensaje "ALGO TE FRENA" (la oscuridad), no permitiéndonos el acceso al lugar.

Bien, ya basta de pistas. Antes de meternos de lleno en el estudio del manejo del programa, aceptemos un último consejo: los programas de aventura precisan de mucha paciencia e imaginación. No hay que desesperarse por tanto si las dunas del «desierto itinerante», en su incesante viaje, bloquean nuestras posibles salidas. En último caso siempre podemos interrumpir el programa y recomenzar con **RUN.** Recordad: «No es mejor pescador el que tiene la caña más larga, sino el que tiene la paciencia más grande».

Los comandos a utilizar para «actuar» en nuestra historia consisten en una o dos palabras. Habrá que teclear en primer lugar el verbo (es decir, la acción que deseamos realizar), y a continuación (no siempre) nombrar el objeto con el que vamos a realizar dicha acción. En algunos casos el ordenador nos pedirá más información después de haber dado una orden; por ejemplo: si decimos "MATA DRAGON", el Spectrum nos preguntará con qué deseamos destruir a nuestro acérrimo enemigo; en este caso hay que introducir el nombre del arma o herramienta con que deseamos hacerlo, o nada si queremos utilizar nuestras propias manos.

Ante todo hay que introducir todos los comandos correctamente, esto es, bien deletreados. Si alguna letra de la orden a ejecutar estuviese equivocada, el ordenador hará caso omiso, y te presentará en la pantalla el mensaje "NO COMPRENDO", seguido del comando erróneo introducido.

Para ahorrarnos trabajo y evitar errores de escritura, la mayoría de las palabras-comando pueden ser abreviadas a 1, 2 ó 3 letras iniciales. Por otra parte, es conveniente dibujar un mapa de nues-

tro recorrido, para tener una referencia del punto en donde nos encontramos, evitando así nuestra desorientación.

Como pista, y sin que sirva de precedente, diremos que existe un oasis al sureste de la posición inicial, y que las dunas del desierto pueden bloquearte las salidas que hubieses descubierto con anterioridad. En la tabla adjunta se detallan todos los comandos que el ordenador reconoce. Con suerte, paciencia e imaginación, y valiéndose sólo de las palabras-comando reseñados más adelante, podremos llegar a conseguir nuestro objetivo.

La información que el ordenador necesita de cada objeto consiste en su propia descripción: dónde está, qué clase de objeto es (tesoro, monstruo, llave...), cómo es de fuerte, etc... Este sistema permite una mayor flexibilidad para la actuación del programa, según la actitud que el protagonista adopte ante el objeto en cuestión.

Tanto los objetos como los lugares y comandos, se encuentran codificados en el programa mediante un número, que supone un diferente significado en cada caso. Cuando se trata de objetos, éstos pueden poseer tres valores diferentes:

* valor -1: objetos que no se pueden ver o usar, u objetos cogidos con anterioridad.
* valor 0: objetos que el jugador lleva consigo.
* valor 1: resto de los objetos que no son utilizados en ese momento por la situación del jugador.

Los números que han sido asignados a los verbos (coge, deja, come...) son utilizados por el programa para localizar la subrutina que realiza la acción de dicho verbo. La codificación de cada comando se corresponde con la posición que ostenta cada uno en la **DATA**. Así pues, el primer verbo llevará el número uno, y la subrutina de acceso se encontrará a partir de la línea de programa número 1469 + 31 * (número del verbo). Ya sólo resta añadir que el programa tiene asignados unos números claves, los cuales son empleados por el ordenador para localizar el lugar o escenario en el que discurre la acción presente; estos valores comienzan en el uno y van incrementándose hasta llegar al treinta.

```
   1 REM ************************
   2 REM * J.M.MAYORAL SERRANO *
   3 REM ************************
   5 GO SUB 9995
  10 BORDER 1
  12 PAPER 1: INK 9
  13 CLS
  30 LET L=1
  40 GO SUB 900
  50 POKE 23692,255
  51 RESTORE L+LS: READ M$
  52 PRINT 'TE ENCUENTRAS ';M$
  53 PRINT : LET Q$='ESTAS VIENDO: '
  54 GO SUB 2000
  60 BEEP .1,10
  61 PRINT
  65 REM
  66 REM   PROGRAMA PRINCIPAL
  67 REM
  70 GO SUB 1000
  80 PRINT V$;' ';
  90 GO SUB VR+VN*20
 100 IF F=0 THEN  GO TO 60
 110 PRINT
 111 GO TO 50
 150 REM INIC.VARIABLES PEC.
 900 RESTORE OS
 901 READ OC
 902 DIM O(OC)
 910 FOR I=1 TO OC
 911 READ O(I)
 912 NEXT I
 920 RESTORE VS
 921 READ VC
 930 LET BE=0: LET LU=0
 931 LET GU=0
 932 LET E$='EN UN DESIERTO SIN FIN'
 933 LET P$='EN UN PASADIZO'
 940 RETURN
1000 INPUT 'DIME QUE HAGO ''' LINE S$
1001 IF LEN S$=0 THEN  GO TO 1000
1010 LET A$=''
1011 LET N$=''
1020 LET X=0
1021 FOR I=1 TO LEN S$
1030 LET I$=S$(I TO I)
1040 IF I$>='A' AND I$<='Z' THEN  LET I$=CHR$ (32+COD
E I$)
1050 IF I$=' ' AND X=1 THEN  LET I=LEN S$: GO TO 1080
1051 IF I$=' ' THEN  LET X=1: GO TO 1080
1060 IF X=0 THEN  LET A$=A$+I$
1070 IF X=1 THEN  LET N$=N$+I$
1080 NEXT I
1081 LET W$=A$
1082 LET C=VC
1083 LET D=VS+1
1084 GO SUB 1200
1090 IF VN=0 THEN  PRINT  PAPER 6;' NO COMPRENDO: ';
PAPER 1;'  ';A$: GO TO 1000
1100 LET V$=Y$
1101 RETURN
1200 LET VN=0
1210 RESTORE D
1220 FOR I=1 TO C
1221 READ M$
1222 LET X=LEN M$
1223 IF X>LEN W$ THEN  LET X=LEN W$
1240 IF W$( TO X)=M$( TO X) THEN  LET VN=I: LET Y$=M$
: LET I=C
1260 NEXT I
1261 RETURN
1300 IF N$='' THEN  GO TO 1311
1301 LET W$=N$
1302 LET C=OC
1303 LET D=8001
1304 GO SUB 1200
1305 IF VN=0 THEN  GO TO 1311
1310 PRINT Y$: GO TO 1999
1311 PRINT  PAPER 6;' NO COMPRENDO ': PAPER 1;N$
1312 GO TO 1998
1400 REM VALORES/DIRECCIONES
1500 LET D=1: GO TO 1601
1520 LET D=2: GO TO 1601
1540 LET D=3: GO TO 1601
1560 LET D=4: GO TO 1601
1580 LET D=5: GO TO 1601
1600 LET D=6
1601 PRINT : RESTORE L+LS: READ M$: FOR I=1 TO D: REA
D NL: NEXT I
1602 IF NL=0 THEN  PRINT 'NO PUEDES COGER ESE CAMINO'
: GO TO 1998
1603 RESTORE NL+LS: READ M$,S,S,S,S,S,S
1604 READ S
1605 GO SUB 7100+S: IF F=0 THEN  GO TO 1611
1610 IF O(1)=0 THEN  LET L=NL: GO TO 1999
1611 PRINT 'ALGO TE FRENA': GO TO 1998
1620 GO SUB 1300: IF F=0 THEN  RETURN
1621 IF O(VN)<>L THEN  PRINT 'ESO NO ESTA AQUI': GO T
O 1998
1622 RESTORE OZ+VN: READ S: IF S<0 THEN  GO TO 1997
1624 READ S
1625 READ S
1626 GO SUB CS+S: IF F=0 THEN  GO TO CT
1630 IF CX<>0  THEN  LET CX=CX-1: LET O(VN)=0: RETURN
```

A) De avance:
1) NORTE
2) SUR
3) ESTE
4) OESTE
5) ARRIBA
6) ABAJO

B) De acción:
1) COGE
2) DEJA
3) MATA
4) CORTA
5) NADA
6) ABRE
7) CAVA
8) COME
9) SALTA
10) BEBE
11) AYUDA
12) INVENTARIO
13) PUNTOS

¡!

El programa carece de gráficos definidos, debido a lo cual no aparecerá en el listado ningún carácter subrayado.

*

La grabación del programa se debe llevar a cabo mediante el comando **SAVE**. Si utilizamos el sistema de autoejecución, ésta ha de comenzar a partir de la línea 1. **(SAVE "AVENTURA" LINE 1).**

*

Toda nuestra aventura transcurre en el fantástico «desierto itinerante», por tanto no nos debe extrañar que el camino por el que hemos accedido a determinado lugar, se encuentre poco más tarde bloqueado por una duna.

```
1631 PRINT 'LLEVAS DEMASIADAS COSAS': GO TO 1998
1640 GO SUB 1300: IF F=0 THEN  RETURN
1641 IF O(VN)<>0 THEN  PRINT 'NO LO TIENES': GO TO 19
98
1642 RESTORE OZ+VN: READ S
1643 LET O(VN)=L
1644 IF L=1 AND S<>0 THEN  LET SC=SC+S: LET O(VN)=-1:
 PRINT 'EL ';Y$;' HA DESAPARECIDO'
1645 LET CX=CX+1: IF SC=220 THEN  GO TO 9980
1646 RETURN
1660 GO SUB 1300: IF F=0 THEN  RETURN
1661 LET T$=Y$: LET MN=VN: RESTORE OZ+VN: READ S,MS:
IF S>=0 THEN  GO TO 1997
1662 LET S=0: BEEP .2,25: INPUT 'CON QUE VAS A MATAR
?'''U$: IF U$='' THEN  GO TO 1670
1663 LET C=OC: LET D=OS+1: GO SUB 1200: IF VN=0 THEN
 GO TO 1662
1664 IF O(VN)<>0 THEN  GO TO 1662
1665 RESTORE OZ+VN: READ S,S: IF S=0 THEN  GO TO 1662
1666 PRINT ' CON ';Y$;
1670 PRINT : LET S=S+ST: IF MS>S+RND*20 THEN  GO TO 1
679
1671 IF S>MS+RND*15 THEN  GO TO 1678
1672 PRINT 'EL ';T$;' VUELVE A LA LUCHA': PRINT 'TE S
IENTES DEVIL'
1673 LET ST=ST-RND*5: GO TO 1999
1678 PRINT 'TE HAS CARGADO AL ';T$: PRINT 'EL CUERPO
SE DESVANECE EN UNA   NUBE DE HUMO': LET O(MN)=-1: GO
 TO 1999
1679 PRINT 'EL ';T$;' TE HA MATADO': GO TO 9990
1680 LET L1=L: LET L=0: LET Q$='ESTAS LLEVANDO': GO S
UB 2000: LET L=L1: GO TO 1999
1700 PRINT SC: GO TO 1999
1720 GO SUB 1300: IF F=0 THEN  RETURN
1721 IF VN<>20 OR O(2)<>0 THEN  GO TO 1997
1722 LET O(14)=O(VN): LET O(VN)=-1: PRINT 'UNA LLAVE
DE MARFIL                CAE DEL TECHO': RETURN
1740 IF L<>11 AND L<>29 THEN  GO TO 1997
1741 IF L=11 AND O(3)<>0 THEN  PRINT 'TE CAISTE AL LA
GO                Y TE HAS AHOGADO': GO TO 9990
1742 IF L=11 THEN  LET L=29: GO TO 1999
1743 LET L=11: GO TO 1999
1760 GO SUB 1300: IF F=0 THEN  LET O(22)=-1: LET O(23
)=L: LET LU=1: RETURN
1762 IF L=13 AND O(14)=0 THEN  LET O(24)=-1: LET O(25
)=L: LET GU=1: RETURN
1763 GO TO 1997
1780 IF O(16)<>0 OR L<3 OR L>10 THEN  GO TO 1997
1781 PRINT 'ENCONTRASTE';
1782 IF L=6 AND O(4)=-1 THEN  LET O(4)=L: PRINT ' ALG
O': GO TO 1999
1783 PRINT ' NADA': GO TO 1999
1800 GO SUB 1300: IF F=0 THEN  RETURN
1801 RESTORE OZ+VN: READ S,S1,S1
1802 IF S<0 THEN  GO TO 1997
1803 GO SUB CS+S1: IF F=0 THEN  GO TO 1997
1804 PRINT 'ESTA ALGO DURO PERO...': LET O(VN)=-1: RE
TURN
1820 GO SUB 1300: IF F=0 THEN  RETURN
1821 IF L<>24 OR VN<>7 THEN  GO TO 1830
1822 PRINT 'UN PUENTE APARECE SOBRE          EL PRECIP
ICIO': LET O(15)=L: LET BE=1: GO TO TR
1830 PRINT 'NO PASA NADA': GO TO 1998
1840 IF L=24 OR L=26 THEN  PRINT 'EN LA CAIDA TE HAS
DESNUCADO': GO TO 9990
1841 GO TO 1830
1860 GO SUB 1300: IF F=0 THEN  RETURN
1861 IF VN<>18 THEN  GO TO CT
1862 PRINT 'AQUELLO ERA DELICIOSO': LET O(19)=O(VN):
LET O(VN)=-1: RETURN
1997 PRINT 'NO PUEDES'
1998 LET F=0: RETURN
1999 LET F=1: RETURN
2000 LET X=0: PRINT Q$
2010 RESTORE OS+1: FOR I=1 TO OC: READ M$
2020 IF L<>O(I) THEN  GO TO 2030
2021 LET X=1: PRINT : PRINT '  UN/A';
2024 PRINT ' ';M$
2030 NEXT I
2040 IF X=0 THEN  PRINT '  NADA'
2050 RETURN
3000 REM BASE/DATOS DIRECCIONES
7000 DATA 19
7001 DATA 'norte','sur','este','oeste','arriba','abaj
o'
7002 DATA 'coge','deja','mata','inventario','puntos'
7003 DATA 'corta','nada','abre','cava'
7004 DATA 'come','ayuda','salta','bebe'
7100 GO TO 1999
7101 GO TO 1998+LU
7102 GO TO 1998+GU
7103 GO TO 1998+BE
7104 GO TO 1998+(O(3)=0)
7105 GO TO 1998+(O(11)=-1)
7106 GO TO 1998+(O(12)=-1)
7107 GO TO 1998+(O(10)=-1)
7150 IF O(18)<>0 THEN  GO TO 1999
7151 LET O(18)=-1: LET O(12)=-1: PRINT 'AL ENTRAR,UN
PIRATA TE ROBA EL RON Y HUYE.': GO TO 1999
7160 PRINT 'EL DIABLO TE MATO'
7161 GO TO 9990
7165 REM BASE/DATOS PROG.PRINC.
8000 DATA 25,1,12,27,-1,14,17,21,25,29,12,14,17,20,-1
,-1,11,2,2,-1,19,11,1,-1,13,-1
8001 DATA 'lampara de metal'
8002 DATA 'espada'
8003 DATA 'daga'
8004 DATA 'llave'
8005 DATA 'alfombra persa'
8006 DATA 'moneda de oro'
8007 DATA 'vara de plata'
8008 DATA 'ruby'
8009 DATA 'diamante'
8010 DATA 'gigante'
8011 DATA 'dragon verde'
8012 DATA 'pirata'
8013 DATA 'diablo'
8014 DATA 'llave de marfil'
8015 DATA 'puente de cristal'
8016 DATA 'pala'
8017 DATA 'mesa'
8018 DATA 'botella de ron'
8019 DATA 'botella vacia'
8020 DATA 'soga que cuelga del techo'
8021 DATA 'gran lago'
8022 DATA 'puerta cerrada'
8023 DATA 'puerta'
8024 DATA 'verja cerrada con candado'
8025 DATA 'verja'
8501 DATA 0,0,0
8502 DATA 0,20,7
8503 DATA 0,0,0
8504 DATA 0,0,0
8505 DATA 10,0,5
8506 DATA 50,0,6
8507 DATA 20,0,0
8508 DATA 30,0,0
8509 DATA 100,0,0
8510 DATA -1,87,0
8511 DATA -1,110,0
8512 DATA -1,210,0
8513 DATA -1,210,0
8514 DATA 10,0,0
8515 DATA 0,0,0
8516 DATA 0,0,0
8517 DATA 0,0,-5102
8518 DATA 0,0,0
8519 DATA 0,0,0
8520 DATA 0,0,-5102
8521 DATA 0,0,-5102
8522 DATA 0,0,-5102
8523 DATA 0,0,-5102
8524 DATA 0,0,-5102
8525 DATA 0,0,-5102
9001 DATA 'EN EL CUARTO DE ESTAR DE UNA CASA,UN LETRE
RO DICE:DEPOSITA TODOS LOS TESOROS AQUI',2,0,0,0,0,12
,0
9002 DATA 'EN LA COCINA',Z,1,3,Z,Z,Z,Z,Z
9003 DATA E$,0,4,5,2,0,0,0
9004 DATA E$,3,7,6,0,0,0,0
9005 DATA E$,0,6,7,3,0,0,0
9006 DATA E$,5,8,0,4,0,0,0
9007 DATA E$,0,0,0,4,0,0,0
9008 DATA E$,6,9,0,0,0,0,0
9009 DATA E$,8,0,10,0,0,0,0
9010 DATA E$,11,0,0,9,0,0,0
9011 DATA 'UN OASIS',0,10,0,0,0,0,0
9012 DATA 'EN UNA CUEVA',0,16,13,0,1,0,1
9013 DATA 'A LAS PUERTAS DEL INFIERNO',0,0,0,12,0,20,
0
9014 DATA 'EN UNA OSCURISIMA CAVERNA',0,15,0,0,0,0,0
9015 DATA P$,14,18,0,16,0,0,0
9016 DATA P$,12,0,15,0,0,17,0
9017 DATA 'EN LA GUARIDA DE LOS PIRATAS',0,0,0,0,16,0
,50
9018 DATA P$,15,19,0,0,0,0,0
9019 DATA P$,18,0,0,0,0,0,0
9020 DATA 'EN EL INFIERNO... UN DIABLO DICE: !ENCUENT
RA EL CAMINO CORRECTO Y SOBREVIVE SINO MUERE',30,30,3
0,21,30,30,2
9021 DATA P$,0,24,22,20,0,0,0
9022 DATA 'EN UN FINAL MORTIFERO',0,0,0,21,0,0,0
9023 DATA 'EN EL INFIERNO',0,0,0,0,0,0
9024 DATA 'AL BORDE DE UN PROFUNDO ABISMO',21,25,0,0,
0,0,0
9025 DATA 'EN UN BONITO CORREDOR',24,0,27,0,0,0,3
9026 DATA 'AL BORDE DE UN PROFUNDO POZO. HAY INDICIOS
 DE .                    FUEGO Y AZUFRE',0,27,0,0,0,
23,0
9027 DATA P$,26,0,0,25,0,28,0
9028 DATA P$,0,0,0,0,27,18,0
9029 DATA 'NADANDO EN UN GRAN LAGO',0,0,0,0,0,0,4
9030 DATA '',0,0,0,0,0,0,60
9035 REM FIN BASE/DATOS PROG.
9036 REM
9037 REM
9038 REM SBR.FIN JUEGO
9980 FOR N=1 TO 4: PRINT : NEXT N
9981 PRINT  PAPER 1; FLASH 1;' LO CONSEGUISTE'
9990 PRINT : PRINT 'PUNTUACION ';SC: PRINT 'QUIERES I
NTENTARLO DE NUEVO ?'
9991 IF INKEY$='' THEN  GO TO 9991
9992 BEEP .2,20: IF INKEY$='S' THEN  GO TO 5
9993 IF INKEY$='N' THEN  GO TO 10000
9994 GO TO 9991
9995 LET TR=1999: LET FA=1998: LET LS=9000
9996 LET OS=8000: LET OZ=8500: LET VS=7000
9997 LET CS=7100: LET SC=0
9998 LET VR=1480: LET Z=0: LET CT=1997: LET DR=1601:
LET CX=6: LET ST=100
9999 RETURN
```

UNA Y OTRA VEZ

EN los ejemplos que hemos visto hasta ahora se plantea siempre una estructura cíclica, en la que el programa acepta datos del teclado (**INPUT**), efectúa determinados cálculos o evaluaciones con ellos y, por último, produce una salida impresa de los resultados, antes de volver a ejecutarse.

A esta estructura de programa la denominamos ciclo: en ella, las cosas se repiten una y otra vez hasta que no ordenamos lo contrario, con la respuesta a un mensaje de si deseamos continuar o no, o simplemente interrumpiendo la ejecución mediante **BREAK**.

Por supuesto que también podemos dejar que estos programas terminen después de la impresión de los resultados y que, en el caso de que se desee continuar, vuelvan a ponerse en ejecución desde la primera sentencia, por medio del comando **RUN**.

REPITIENDO

Dentro de esta gran estructura cíclica, nos es muy conveniente poder incluir otro tipo de construcciones, a las que denominamos BUCLES, y que resultan de gran utilidad cuando deseamos repetir un bloque de instrucciones un determinado número de veces.

El formato que nos va a permitir codificar estas estructuras de bucle, se basa en tres palabras clave y el nombre de variable que designemos para controlar dicha estructura, adoptando la forma:

FOR variable = valor-inicial TO valor-final STEP incremento
instrucción
...
instrucción
NEXT variable

Supongamos que queremos obtener la suma de los diez primeros números naturales; codificando el programa de la forma que hasta ahora conocemos, podríamos obtener algo como esto:

```
10 REM - SUMA DE NUMEROS
20 LET S=0
30 LET N=0
40 LET N=N+1
50 LET S=S+N
60 IF N<10 THEN GO TO 40
70 CLS
80 PRINT "La suma es:    ";S
```

Aplicando los conocimientos recientemente adquiridos, podríamos ahorrar instrucciones y ganar en claridad escribiendo:

```
10 REM - SUMA DE NUMEROS
20 LET S=0
30 FOR N=1 TO 10
40 LET S=S+N
50 NEXT N
60 CLS
70 PRINT "La suma es:    ";S
```

La estructura del bucle que hemos propuesto se encuentra en las líneas 30 a 50; la primera contiene el nombre de la variable y los valores mínimo y máximo que debe adoptar, así como el incremento por cada nueva pasada. La segunda es la operación que debe realizarse para los diferentes valores de **N**. Por último, la tercera contiene el final del bucle, especificando el nombre de la variable que lo controla.

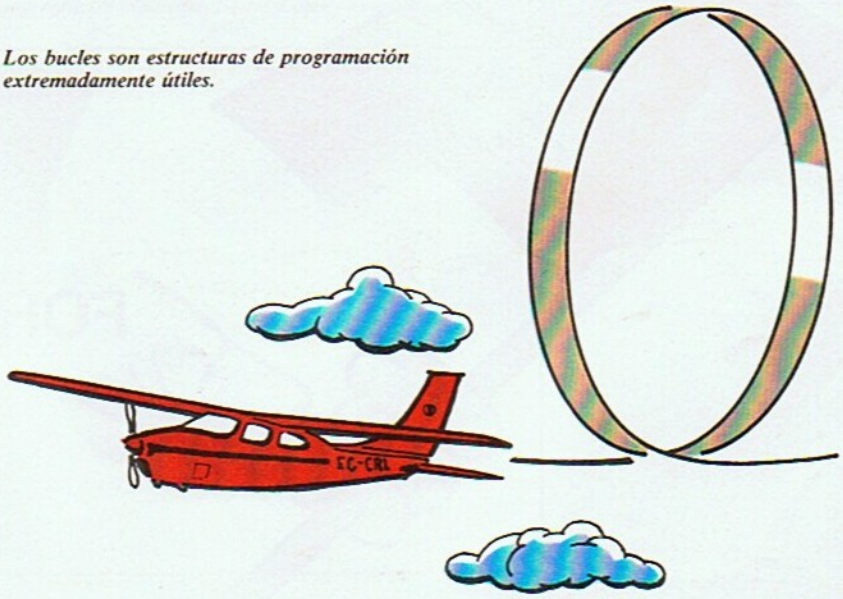
¡!

Al agrupar instrucciones en una misma línea, las sentencias de bifurcación sólo pueden dirigirse al grupo completo.

*

No existe limitación en cuanto a los topes inferior y superior marcados para el **FOR**, ni en cuanto al **STEP** o paso que se fija. Estos pueden ser números enteros, fraccionarios, negativos o decimales, así como variables.

Los bucles son estructuras de programación extremadamente útiles.



Los bucles **FOR NEXT** no pueden entremezclarse. O bien se encuentra uno totalmente en el interior del otro, o son independientes.

Con el signo de puntuación dos puntos (:), pueden codificarse cuantas instrucciones se deseen agrupadas bajo un mismo número de línea.

En nuestro ejemplo, el programa va a recorrer las instrucciones 30 a 50 diez veces, siendo los respectivos valores de **N** en cada pasada 1, 2, 3, 4 ... 10. Como el caso más normal en los bucles **FOR-NEXT** es que el incremento de la variable sea 1, puede omitirse el **STEP** cuando el incremento sea la unidad: **30 FOR N=1 TO 10**, equivaldría por tanto a **30 FOR N=1 TO 10 STEP 1**. Lo que realmente hace el ordenador cuando se encuentra con un bucle de este tipo, es muy parecido al primer ejemplo de programa propuesto por nosotros: primero asigna a la variable del **FOR** el valor mínimo y ejecuta todas las instrucciones que siguen hasta encontrar el **NEXT**, entonces suma el **STEP** a la variable, comparando si ésta es mayor que el valor máximo. En caso de que lo sea, se sale del bucle y continúa con la siguiente instrucción en secuencia, en nuestro ejemplo la 60. Si resulta ser menor o igual, efectúa una bifurcación al comienzo del bucle, inmediatamente después del **FOR**.

Para practicar un poco, obtendremos ahora la suma de los 50 primeros números pares (2550):

```
10 REM - SUMA DE NUMEROS PARES
20 LET S=0
30 FOR N=2 TO 100 STEP 2
40 LET S=S+N
50 NEXT N
60 CLS
70 PRINT "La suma de pares es:  ";S
```

Esta vez es absolutamente necesario especificar el incremento de la variable (**STEP**), puesto que es diferente de la unidad. También hemos podido ver lo fácil que resulta adaptar el programa utilizando esta estructura de bucle.

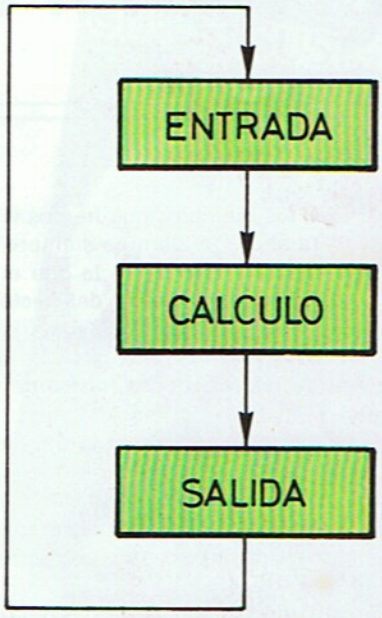
Una de las pocas restricciones que tiene la estructura **FOR-NEXT**, es que el nombre de la variable que sirve de contador debe contener un solo carácter, es decir, no son válidas como variables de bucle las compuestas por más de una letra. Lejos de representar un inconveniente, debemos decir que es práctica común entre programadores el uso de variables de **FOR** de un solo carácter, y preferentemente: **I, J, K** ... Las variables de control de **FOR** se suelen denominar también INDICES.

Por lo demás, no existe limitación alguna en cuanto a los topes inferior y superior marcados en el **FOR**, ni en cuanto al **STEP** o paso que se fija. Estos pueden ser números enteros, fraccionarios, negativos o decimales, así como variables. Algunos ejemplos válidos son:

```
FOR I=9 TO -9 STEP -1
FOR I=3/4 TO 3/8 STEP -.001
FOR I=0 TO 99 STEP .001
FOR I=A TO B STEP C
```

Otra característica es que no es necesario que el límite superior sea rebasado exactamente por la unidad o por la cantidad especificada como **STEP** para que el bucle llegue a su fin, sino que basta con que el valor adquirido por la variable de control en el incremento (**NEXT**) sea mayor que el límite superior, cuando el paso (**STEP**) es positivo. De forma análoga, cuando el paso es negativo, se saldrá del bucle al alcanzar un valor que sea menor. Veamos un ejemplo:

Muy frecuentemente, las estructuras de los programas resultan ser cíclicas.



La sentencia **FOR** *es la «señal de partida» de los bucles BASIC.*

La sentencia STEP *sirve para marcar el paso de las variables indice en los bucles* FOR-NEXT.

```
10 FOR I=1 TO 20 STEP 3
20 PRINT I
30 NEXT I
40 PRINT "Valor de salida:  ";I
```

Los sucesivos valores de **I** que serán impresos antes de finalizar el bucle serán:

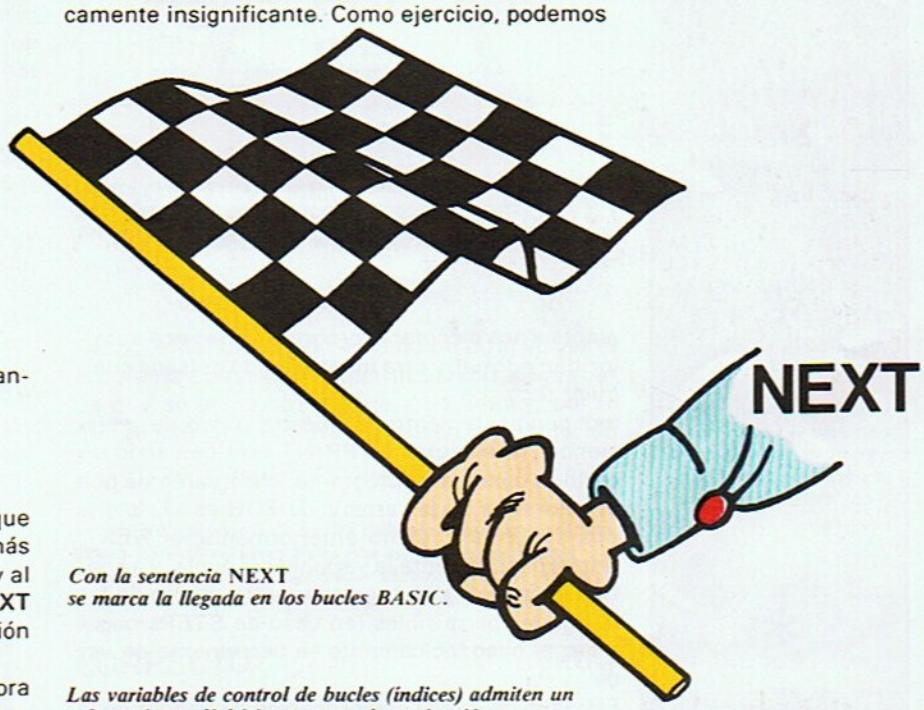
1, 4, 7, 10, 13, 16 y 19

Y el valor final de la variable **I** será **22**, ya que la instrucción **NEXT** la incrementó una vez más para comparar si superaba el límite máximo, y al cerciorarse de ello, salió del bucle **FOR-NEXT** para imprimir el valor final de **I** en la instrucción 40.

Como aplicación de lo dicho, simularemos ahora la cuenta atrás para el lanzamiento de una nave espacial, que en atención a los más impacientes, será tan solo de 20 segundos:

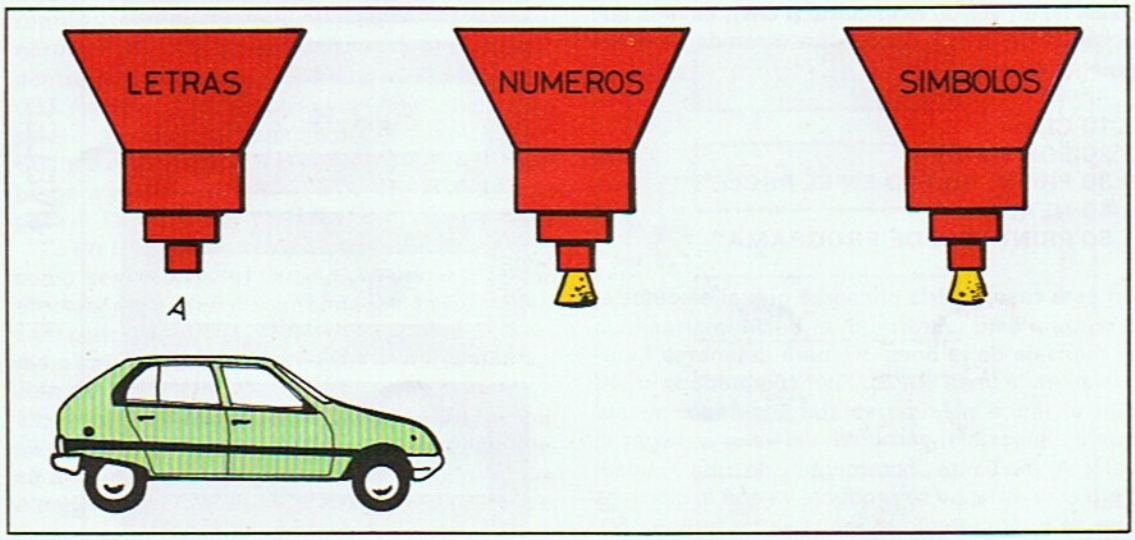
```
10 REM - CUENTA ATRAS
20 FOR I=20 TO 0 STEP -1
30 CLS
40 PRINT I
50 PAUSE 50
60 NEXT I
```

La instrucción **PAUSE** con el argumento 50 produce una parada de aproximadamente un segundo, por lo que el efecto quedará bastante logrado, siempre que nadie opte por darle una velocidad fulgurante al conteo, manteniendo pulsada una tecla, lo que convertiría el retardo en prácticamente insignificante. Como ejercicio, podemos



Con la sentencia NEXT *se marca la llegada en los bucles BASIC.*

Las variables de control de bucles (índices) admiten un solo carácter alfabético para su denominación.



¡!

Existen unas estructuras de programa denominadas BUCLES, que nos permiten recorrer un determinado bloque de instrucciones un número concreto de veces.

*

Es práctica común entre programadores, el uso para variables de FOR de: I, J, K ...

*

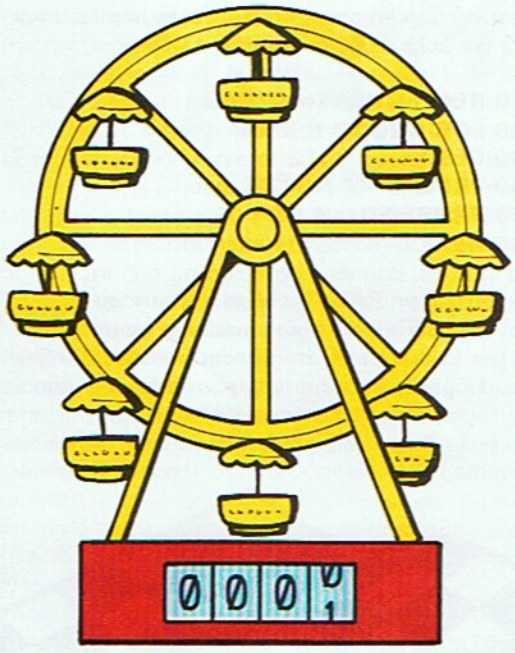
Una restricción de la estructura **FOR NEXT** es que el nombre de la variable de control (índice) debe ser de un solo carácter.

*

El formato que permite codificar las estructuras de bucle, se basa en tres palabras clave y el nombre de variable de control que designemos, adoptando la forma:
FOR variable=valor-inic. TO valor-final STEP incremento

*

Cuando una estructura de bucle incluye a otro o varios se denomina «anidamiento».



Los bucles ejecutan una determinada tarea durante un número de veces especificado por el programador.

plantearnos ejecutar el programa una vez a su velocidad normal, y otra manteniendo pulsada cualquier tecla.
Independientemente de nuestro grado de impaciencia, si ejecutamos **PRINT I** en comando directo al concluir el programa, observaremos que el valor final de la variable de **FOR** es –1, lo que nos confirma lo dicho anteriormente: el **NEXT**, primero incrementa la variable del bucle, y a continuación averigua si se encuentra dentro de los márgenes permisibles (en caso de **STEPs** negativos, el paso lógicamente se decrementa en vez de incrementarse).
El hecho de que la comprobación de validez de la variable de control se realice al salir del bucle **(NEXT)**, en vez de al iniciarlo **(FOR)**, es más importante de lo que parece. Lo veremos en el siguiente ejemplo:

```
10 CLS
20 FOR I=2 TO 1
30 PRINT "ENTRO EN EL BUCLE"
40 NEXT I
50 PRINT "FIN DE PROGRAMA"
```

En este caso, podría pensarse que al ejecutar el programa éste entraría en el bucle imprimiendo el mensaje de la línea 30, para detenerse finalmente en la línea 50, al haber comprobado en 40 que el límite máximo estaba superado, puesto que es imposible, partiendo del valor 2, llegar al valor 1 mediante incrementos de una unidad. Pero este error no se produce, ya que lo cierto es que se imprime sólo el mensaje de la línea 50, porque antes de entrar en el bucle, el intérprete del BASIC Spectrum comprueba si la variable del **FOR** se encuentra ya fuera del rango, y si es así, no ejecuta el bucle. Esta es una comprobación excepcional que el ordenador sólo realiza la primera vez (el resto, como ya sabemos, se efectúan al llegar al **NEXT**).
También nos puede suceder que codifiquemos un bucle infinito, especificando un incremento nulo:

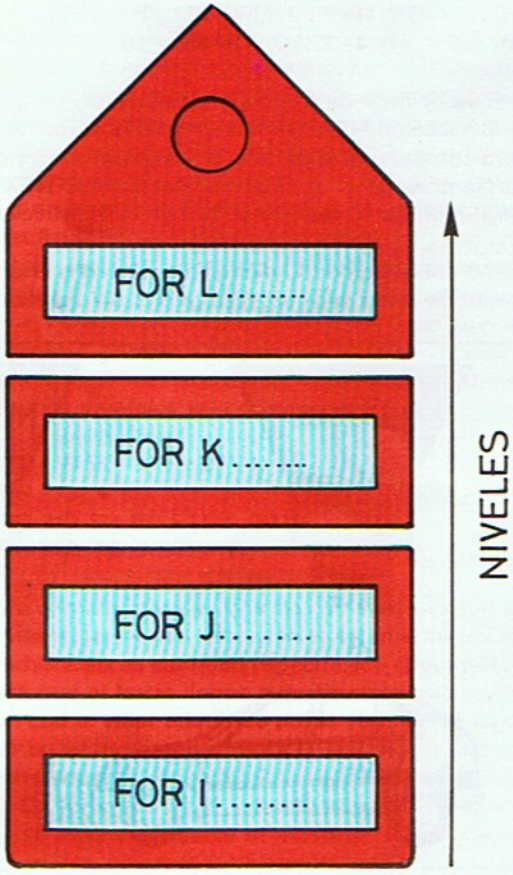
```
10 CLS
20 FOR I=0 TO 0 STEP 0
30 PRINT "OTRA VEZ"
40 NEXT I
```

Lo que realmente se consigue con esto es no emplear la sentencia imperativa **GO TO**; recurriendo a este método de bucle sin fin, el programa anterior produce los mismos resultados que:

```
10 CLS
20 PRINT "OTRA VEZ"
30 GO TO 20
```

Ya que en ambos casos la instrucción **PRINT** se repite de modo indefinido y la única forma de sa-

Cada paso en el anidamiento de bucles se conoce como NIVEL.



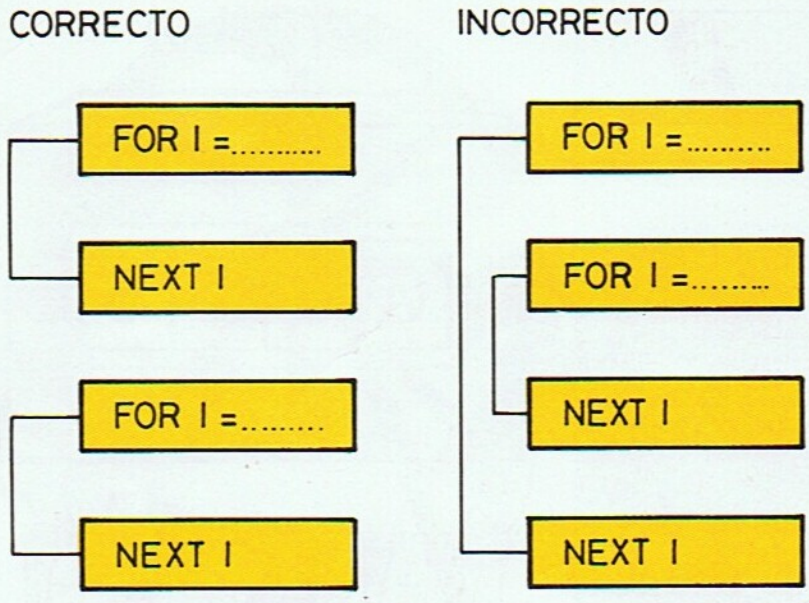
lir de este embrollo es mediante la pulsación de **CAPS SHIFT** y **SPACE (BREAK)**.

RIZANDO EL RIZO

También existe la posibilidad de construir estructuras más complejas, incluyendo uno o más bucles dentro de otro. A este sistema de programación se le denomina ANIDAMIENTO de bucles, y tiene como restricción que, lógicamente, los bucles no pueden entremezclarse, es decir, o bien se encuentra uno totalmente en el interior del otro (diferente nivel), o bien son totalmente independientes (del mismo nivel).
En el primer caso, las variables de control elegidas deben ser obligatoriamente distintas. En el segundo, pueden serlo o no, ya que al ser los bucles independientes es necesario que termine uno para que comience el siguiente, por lo que no puede existir problema alguno.
Veamos ahora un ejemplo de anidamiento de bucles, escribiendo una lista de números desde el 101 al 276, distribuida en 22 líneas de 8 números:

```
10 REM - LISTA DE NUMEROS
20 CLS
30 FOR I=101 TO 276 STEP 8
40 FOR J=I TO I+7
50 PRINT "  ";J;
60 NEXT J
70 NEXT I
```

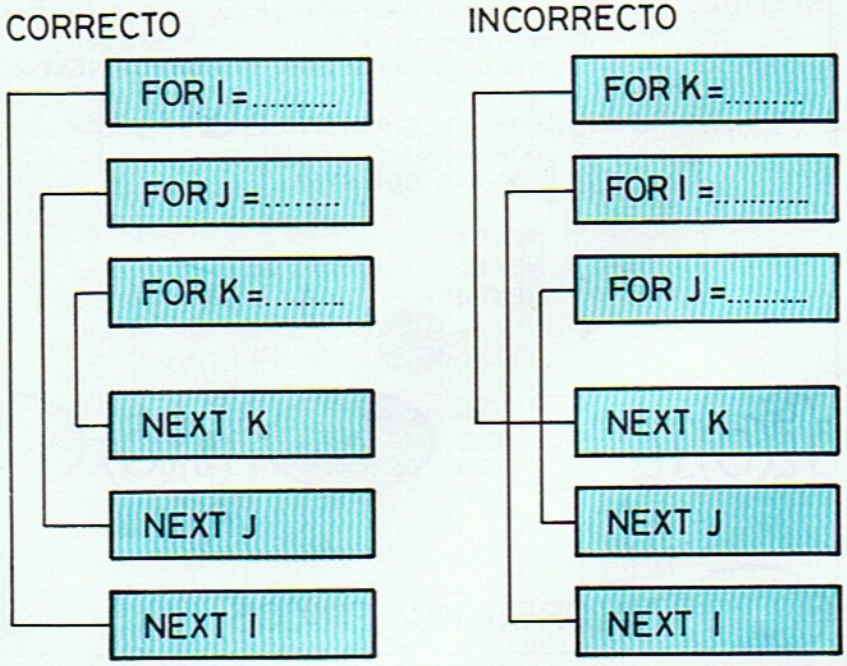
Debemos fijarnos en que los bucles son de diferentes variables **I** y **J**, y que el bucle **J** es de menor nivel que el **I**, ya que se encuentra contenido dentro de él, al estar el **NEXT** de **J** antes que el de **I**. Esta es la única estructura válida y correcta para codificar bucles anidados.
Cuando el programa alcanza la instrucción 30 por primera vez, **I** toma el valor 101. A continuuación pasa a la instrucción 40 donde **J** toma el valor de **I**. En la instruccion 50 se imprime un espacio como separador y el valor de la variable **J**, sin efectuar el retorno de carro, dado que la instrucción concluye con un punto y coma (;). El programa alcanza entonces por primera vez la instrucción 60, donde suma 1 a **J** pasando a valer 2. Este momento lo aprovecha para comprobar que se encuentra dentro de límites, por lo que bifurca a la instrucción siguiente a la 40 para volver a imprimir, repitiéndose este bucle hasta 8 veces (la primera y siete más).



Nunca podremos utilizar una misma variable de control para dos bucles anidados; sin embargo, no existirá problema en emplearla en estructuras independientes.

Después del último incremento de **J**, al volver el programa a pasar por la instrucción 60, comprueba que se encuentra ya fuera de límites, por lo que se sale del bucle de **J** y cae en la instrucción

Si en una estructura de bucles anidados unimos mediante líneas las instrucciones **FOR** *con sus correspondientes* **NEXT**, *nunca se deberán cruzar dichas líneas; de no ser así, el anidamiento se habrá efectuado de manera incorrecta.*



¡!

El bucle llega a su fin cuando el valor de la variable del **FOR** en el incremento es mayor que el límite superior, cuando el paso (**STEP**) es positivo. Cuando el paso es negativo, sale del bucle al alcanzar un valor menor.

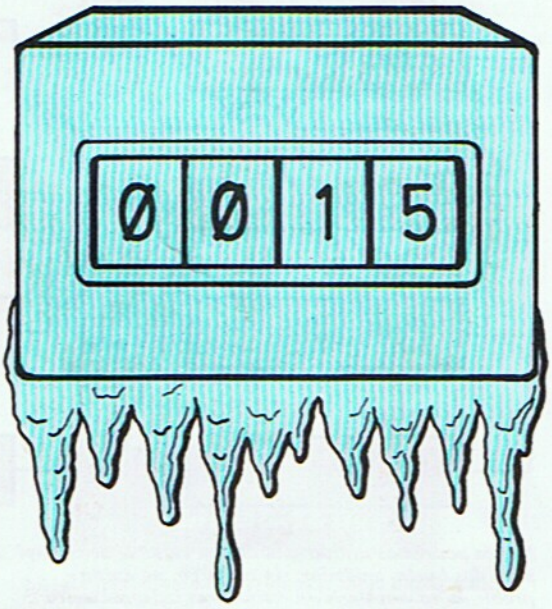
*

La instrucción **PAUSE** con el argumento 50 produce una parada de aproximadamente un segundo.

*

Podemos codificar un bucle infinito, especificando un incremento nulo, con lo que se consigue prescindir de la sentencia imperativa **GO TO**.

STEP Ø



El STEP 0 bloquea la acción de NEXT, constituyendo bucles sin fin.

70 que incrementa **I** (el bucle exterior), y se comprueba que el valor de **I** se encuentra aún dentro de límites, por lo que se transfiere la ejecución del programa a la línea siguiente a la 30, donde vuelve a comenzar el bucle de **J** (el de menor nivel).

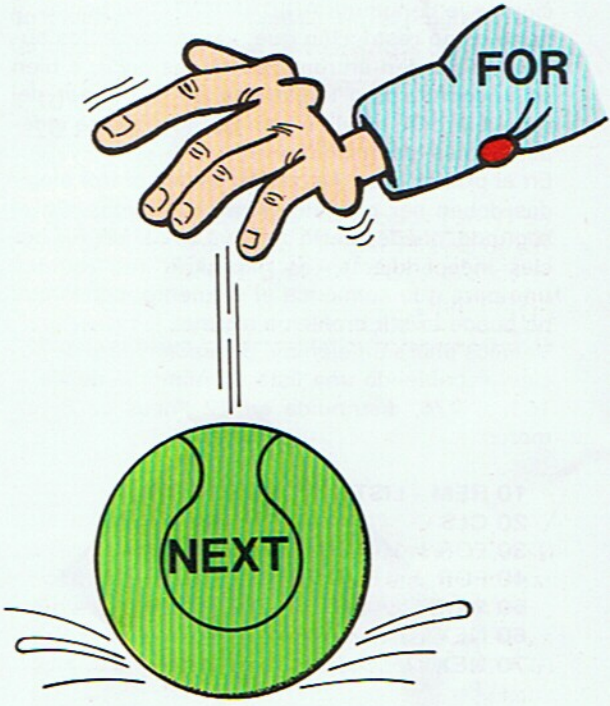
Este estado de cosas seguirá repitiéndose hasta que **J** primero e **I** después, sobrepasen los límites superiores fijados en sus correspondientes instrucciones **FOR**, en las líneas 30 y 40.

Una estructura no válida de bucle **FOR NEXT** es:

```
FOR I=A TO B STEP C
FOR J=D TO E STEP F
FOR K=G TO H STEP L
...
NEXT K
NEXT I
NEXT J
```

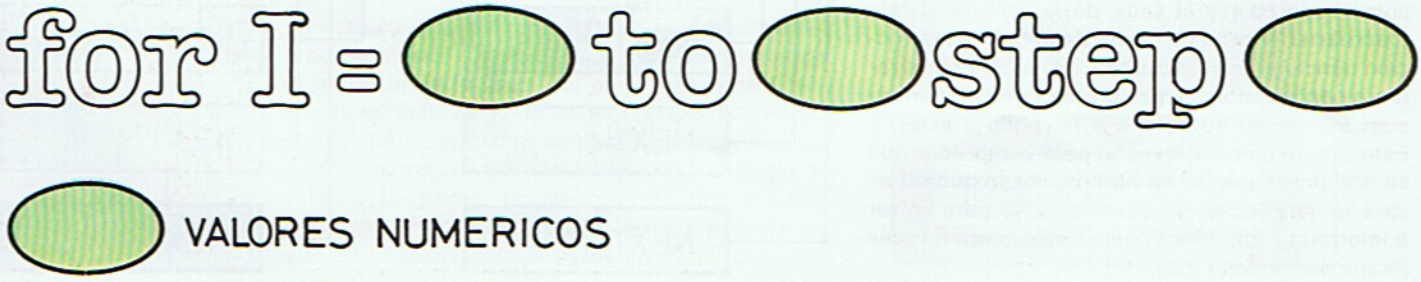
En este bucle está alterado el orden lógico de las instrucciones **NEXT**. Ya que el bucle de la variable **J** es interior (menor nivel) al de la variable **I**; la posición de los **NEXT** debe ser la contraria. Cuando los bucles están anidados (uno dentro de otro), siempre deben cerrarse primero los bucles más interiores y después los exteriores. La estructura siguiente es, por tanto, la correcta:

```
FOR I=A TO B STEP C
FOR J=D TO E STEP F
FOR K=G TO H STEP L
...
NEXT K
NEXT J
NEXT I
```



En las instrucciones FOR se inicializa el número de veces que deseamos se ejecute el bucle correspondiente.

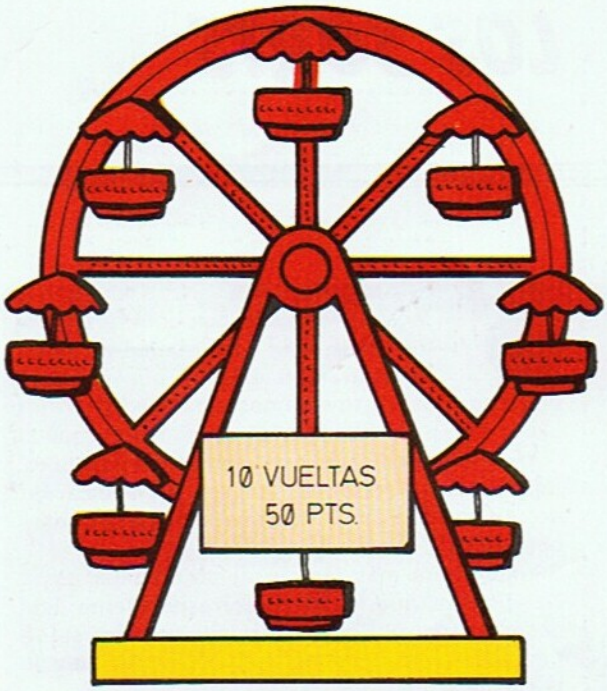
Tanto para los límites como para el paso de una estructura de bucle, habremos de emplear forzosamente valores numéricos.



Otra estructura también correcta es la de bucles completamente separados uno de otro, es decir, no anidados:

```
FOR I=A TO B STEP C
...
NEXT I
FOR J=D TO E STEP F
...
NEXT J
FOR K=G TO H STEP L
...
NEXT K
```

Por supuesto, los resultados que se obtienen en un caso y otro son completamente diferentes, por lo que el programador debe elegir en cada momento la estructura que le conviene. Con todo ello, la idea que debe quedar clara es la de codificar los bucles correctamente, de modo que no se entrecrucen, bien de forma anidada o completamente independientes, según las necesidades.



La ejecución de un bucle supone la realización repetida de una determinada tarea; de ahí que este tipo de estructuras sea también conocido bajo el nombre de ITERATIVAS.

MAS DE UNA INSTRUCCION POR LINEA

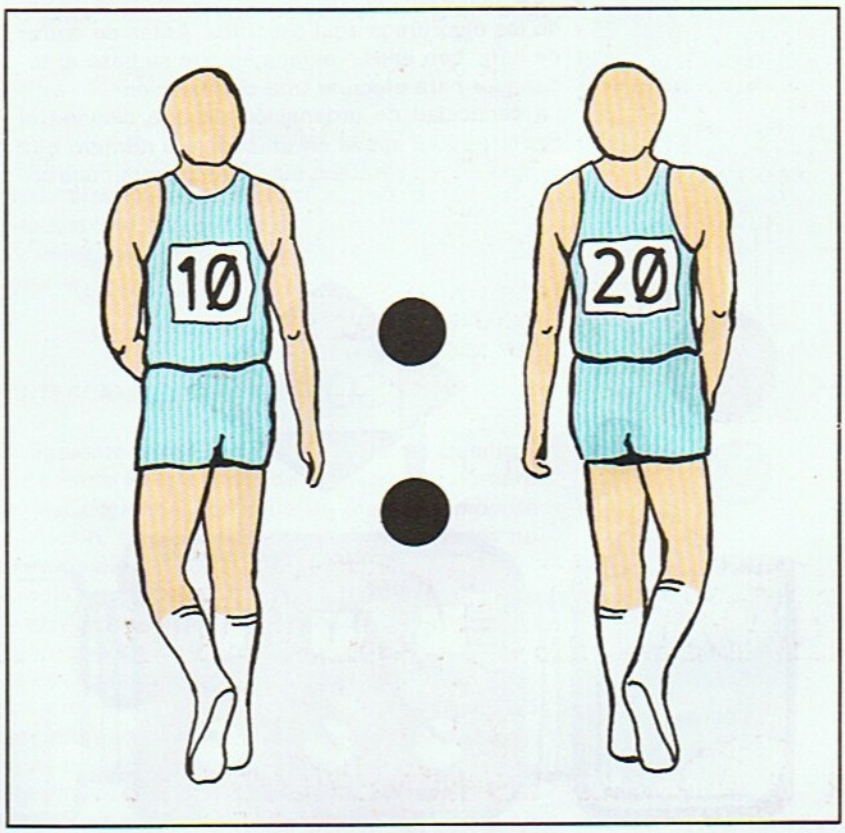
Hasta ahora hemos visto programas en los que cada línea está compuesta por una sola instrucción BASIC. Con el signo de puntuación dos puntos (:), pueden codificarse cuantas instrucciones se deseen, agrupadas bajo un mismo número de línea.

La única restricción consiste en que al agrupar instrucciones bajo un mismo número de línea, las sentencias de bifurcación sólo pueden dirigirse al grupo completo, y no a una concreta de ellas, por lo que las sentencias que agrupemos se ejecutarán siempre en bloque; todas o ninguna:

```
10 REM - SUMA DE NUMEROS
20 LET S=0:LET N=0
30 LET N=N+1:LET S=S+N
40 IF N<10 THEN GO TO 30
50 CLS:PRINT "La suma es:  ";S
```

En este caso, no pueden agruparse las instrucciones de la línea 20 con las de la 30, ya que las primeras se ejecutan una única vez y las de la línea 30 diez veces, a causa de la bifurcación imperativa **GO TO** de la línea 40. Las agrupaciones de las líneas 20, 30 y 50, son perfectamente válidas y nos ahorran líneas de programa.

El separador dos puntos (:) se emplea para combinar varias instrucciones en una misma línea.



LOS SORTS

SON muchas las ocasiones en las que nos enfrentamos con enormes listas de datos, y se hace realmente difícil encontrar en ellas el que estamos buscando. Si los tuviéramos clasificados en un cierto orden, la labor sería mucho más sencilla. A lo mejor, se trata de una colección de nombres, una lista de direcciones, o tal vez, la clasificación de la Liga de Fútbol, con sus equipos y puntos obtenidos.

Siempre será más fácil y rápido manejar datos clasificados, que los de la correspondiente lista desordenada; el Spectrum puede convertirse en una gran ayuda para solucionar este problema de la clasificación.

Los programas que gestionan este trabajo se denominan clasificadores o SORTs, y no se puede decir que unos sean netamente mejores que otros; simplemente, cada cual ofrece mayor interés en determinados casos.

A continuación expondremos los fundamentos de ocho sistemas de SORT y en las páginas 122 y 123, podremos encontrar la traducción al BASIC de los algoritmos aquí descritos. Antes de entrar de lleno con ellos, veamos en qué se basa el ordenador para efectuar una clasificación.

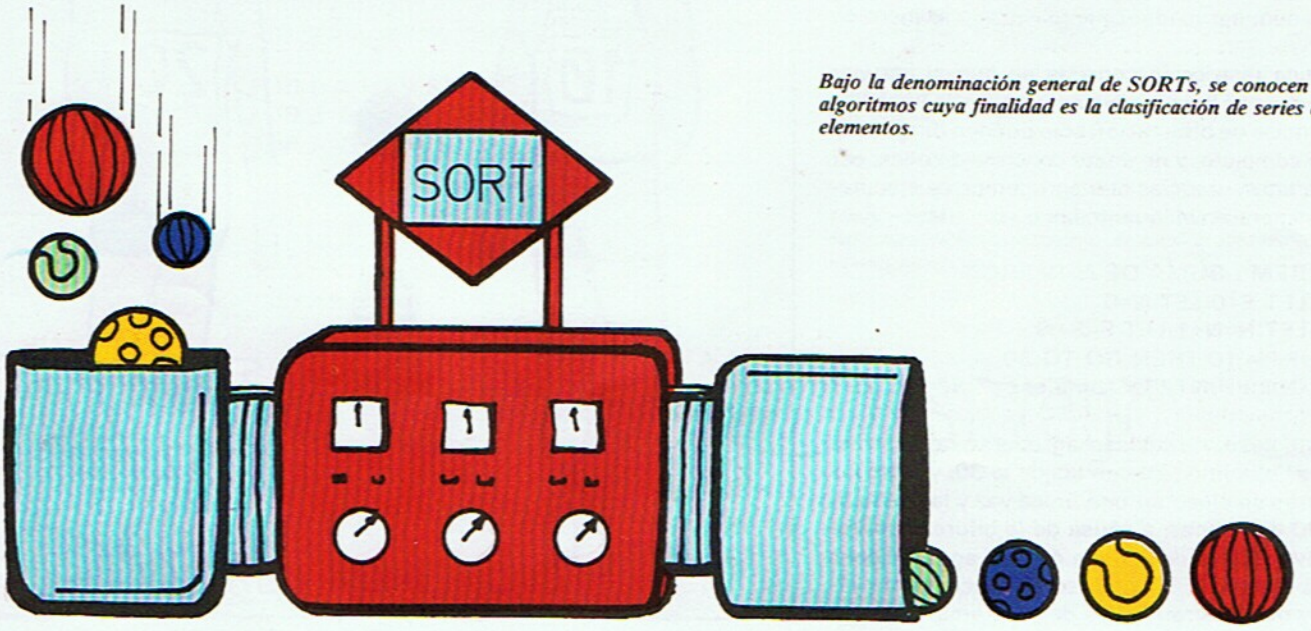
La capacidad de ordenación de que dispone el Spectrum, se apoya en un código o número que asigna a cada carácter; este es el denominado código A.S.C.I.I. *(American Standard Code for Information Interchange,* Código Estandarizado Americano para Intercambio de Información), en el que todos los dígitos, letras o caracteres especiales de los que dispone nuestro Spectrum, llevan asociados un número, entre 0 y 255 (su código A.S.C.I.I.).

Concretamente, los dígitos del 0 al 9 tienen un código entre 48 y 57, las letras mayúsculas entre 65 y 90, las minúsculas entre 97 y 122, y los caracteres especiales, signos de puntuación y símbolos ocupan el resto de los códigos (del 0 al 47 y del 123 al 255). Así por ejemplo, la letra A (código 65) se considera menor que la B (código 66), etc...

Por tanto, podemos decir que el Spectrum lleva a cabo las clasificaciones por «orden alfabético», aunque, eso sí, según «su alfabeto». De todas formas, el código A.S.C.I.I. ha sido dispuesto de manera que coincida con el alfabeto humano en la zona dedicada a las letras. Pasemos ahora a analizar los diferentes sistemas de SORT que podemos utilizar:

* BINARIO o BURBUJA:

Quizá sea el más sencillo de todos los que vamos a tratar. Su peculiar denominación (BURBU-



Bajo la denominación general de SORTs, se conocen los algoritmos cuya finalidad es la clasificación de series de elementos.

JA) responde a la manera en que los datos van «desplazándose» por la lista, hasta ocupar su lugar correspondiente.
El método consiste en comparar el primer elemento de la serie con todos los que le siguen. Si éste es más pequeño, conserva su lugar; en caso contrario, cuando en una comparación encuentre un elemento menor, intercambian sus posiciones, siendo el primero, nuevamente, el menor de los dos. Una vez que el menor de todos los elementos se encuentra en primer lugar, repite el proceso comparando el segundo con los demás, luego el tercero y así sucesivamente, hasta completar el trabajo.

* BURBUJA CON MARCA o BUBBLE:

El método anteriormente explicado tiene un inconveniente muy importante. Supongamos que la lista a clasificar se encuentra al empezar totalmente ordenada; a pesar de ello, el sistema binario realizará todas las comparaciones, aunque no necesite efectuar ningún cambio. Esto lo convierte en un método especialmente lento, cuando se trata de organizar colecciones de datos parcialmente ordenadas.
El sistema de BURBUJA CON MARCA, compara cada elemento con el siguiente, en vez de con todos los de la lista. En el caso de que el siguiente sea mayor que su predecesor, se intercambian sus posiciones. De esta manera, el mayor va «cayendo» por la lista, hasta ocupar su lugar. Una marca dentro del SORT indica si se realizó la pasada sin llevar a cabo ningún intercambio, lo cual significa que cada elemento se encuentra en su lugar y la clasificación ha concluido.

* BUSQUEDA DE MAXIMO Y MINIMO

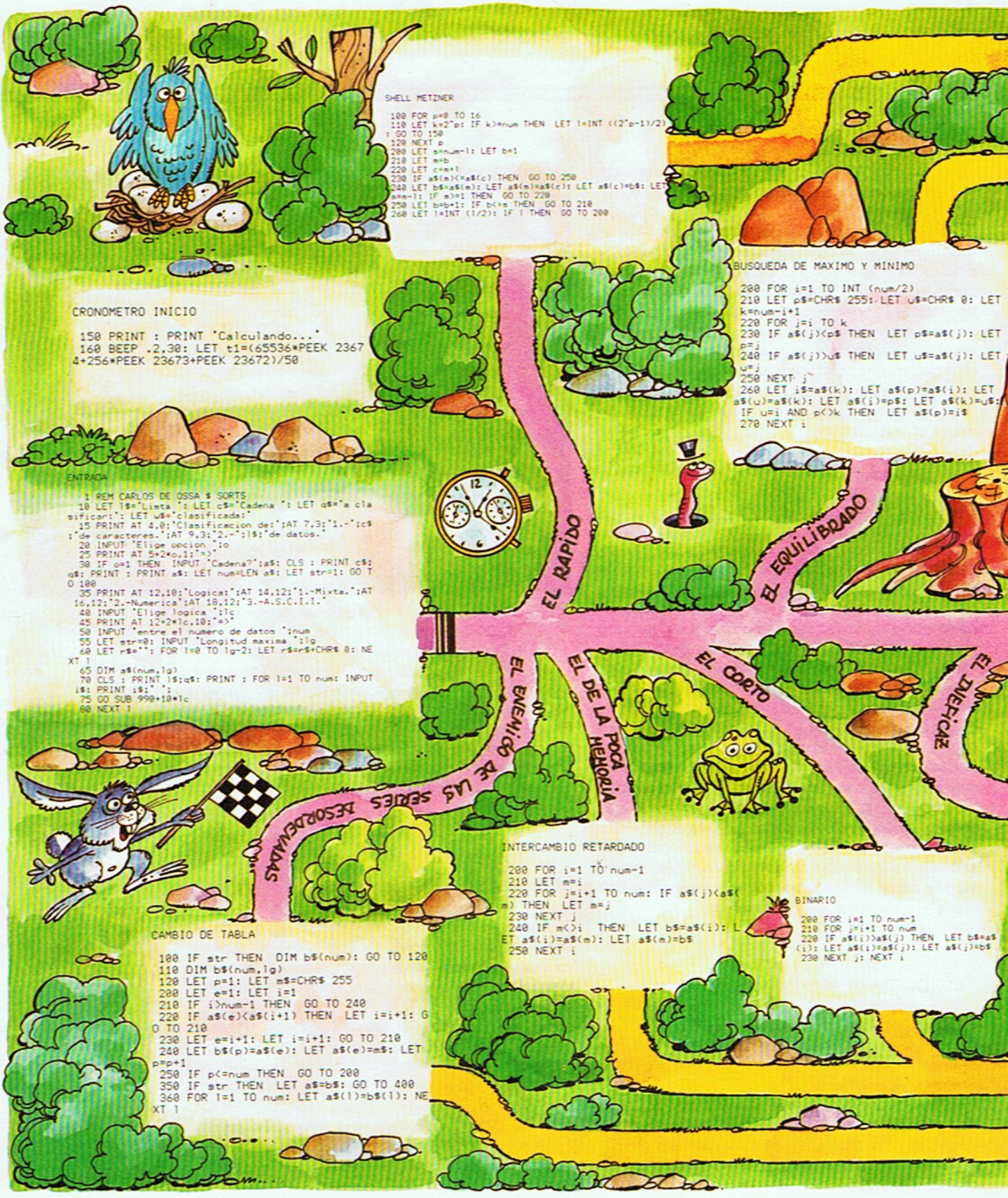
El funcionamiento de este SORT es también sencillo. El método consiste en seleccionar los elementos menor y mayor de la lista y colocarlos en primer y último lugar, respectivamente. A continuación, repetimos la operación para los datos comprendidos entre la segunda y la penúltima posición. Es decir, el bucle principal del clasificador, se reduce en dos elementos cada vez que realizamos una pasada.

Menú del programa ENTRADA para selección de lógica del SORT.

Para listas de datos de igual tamaño, independientemente de si están muy ordenadas o no, el número de pasadas será el mismo, puesto que este sistema no permite determinar si la lista se encuentra ya clasificada, a diferencia del método anterior.

* INTERCAMBIO RETARDADO

El siguiente SORT, es una variante del clasificador binario que, a costa de aumentar el número de instrucciones, consigue un menor tiempo de ejecución, siempre que el volumen de datos no sea muy elevado.
El adjetivo «retardado» responde a que, al efectuar una pasada por la lista, no se realizan intercambios de posición hasta que no hemos encontrado el menor elemento. Para ello, se utiliza un puntero, M en el ejemplo, que señala en cada momento hacia este elemento. Una vez determinado el menor, se coloca en primer lugar y se repite el proceso con los restantes elementos hasta finalizar la clasificación.



SHELL METZNER

```
100 FOR p=0 TO 16
110 LET k=2^p: IF k>=num THEN  LET l=INT ((2^p-1)/2)
: GO TO 150
120 NEXT p
200 LET a=num-l: LET b=1
210 LET m=b
220 LET c=m+l
230 IF a$(m)<=a$(c) THEN  GO TO 250
240 LET b$=a$(m): LET a$(m)=a$(c): LET a$(c)=b$: LET
a=m-l: IF m>=l THEN  GO TO 220
250 LET b=b+l: IF b<=a THEN  GO TO 210
260 LET l=INT (l/2): IF l THEN  GO TO 200
```

BUSQUEDA DE MAXIMO Y MINIMO

```
200 FOR i=1 TO INT (num/2)
210 LET p$=CHR$ 255: LET u$=CHR$ 0: LET
k=num-i+1
220 FOR j=i TO k
230 IF a$(j)<p$ THEN  LET p$=a$(j): LET
p=j
240 IF a$(j)>u$ THEN  LET u$=a$(j): LET
u=j
250 NEXT j
260 LET i$=a$(k): LET a$(p)=a$(i): LET
a$(u)=a$(k): LET a$(i)=p$: LET a$(k)=u$:
IF u=i AND p<>k THEN  LET a$(p)=i$
270 NEXT i
```

CRONOMETRO INICIO

```
150 PRINT : PRINT "Calculando..."
160 BEEP .2,30: LET t1=(65536*PEEK 2367
4+256*PEEK 23673+PEEK 23672)/50
```

ENTRADA

```
1 REM CARLOS DE OSSA $ SORTS
10 LET l$="Lista ": LET c$="Cadena ": LET q$="a cla
sificar:": LET u$="clasificada:"
15 PRINT AT 4,0;"Clasificacion de:";AT 7,3;"1.-";c$
;"de caracteres.";AT 9,3;"2.-";l$;"de datos."
20 INPUT "Elige opcion ";o
25 PRINT AT 5+2*o,1;"=>"
30 IF o=1 THEN  INPUT "Cadena?";a$: CLS : PRINT c$;
q$: PRINT : PRINT a$: LET num=LEN a$: LET str=1: GO T
O 100
35 PRINT AT 12,10;"Logicas:";AT 14,12;"1.-Mixta.";AT
16,12;"2.-Numerica";AT 18,12;"3.-A.S.C.I.I."
40 INPUT "Elige logica ";lc
45 PRINT AT 12+2*lc,10;"=>"
50 INPUT "entre el numero de datos ";num
55 LET str=0: INPUT "Longitud maxima ";lg
60 LET r$="": FOR l=0 TO lg-2: LET r$=r$+CHR$ 0: NE
XT l
65 DIM a$(num,lg)
70 CLS : PRINT l$;q$: PRINT : FOR l=1 TO num: INPUT
i$: PRINT i$;" ";
75 GO SUB 990+10*lc
80 NEXT l
```

INTERCAMBIO RETARDADO

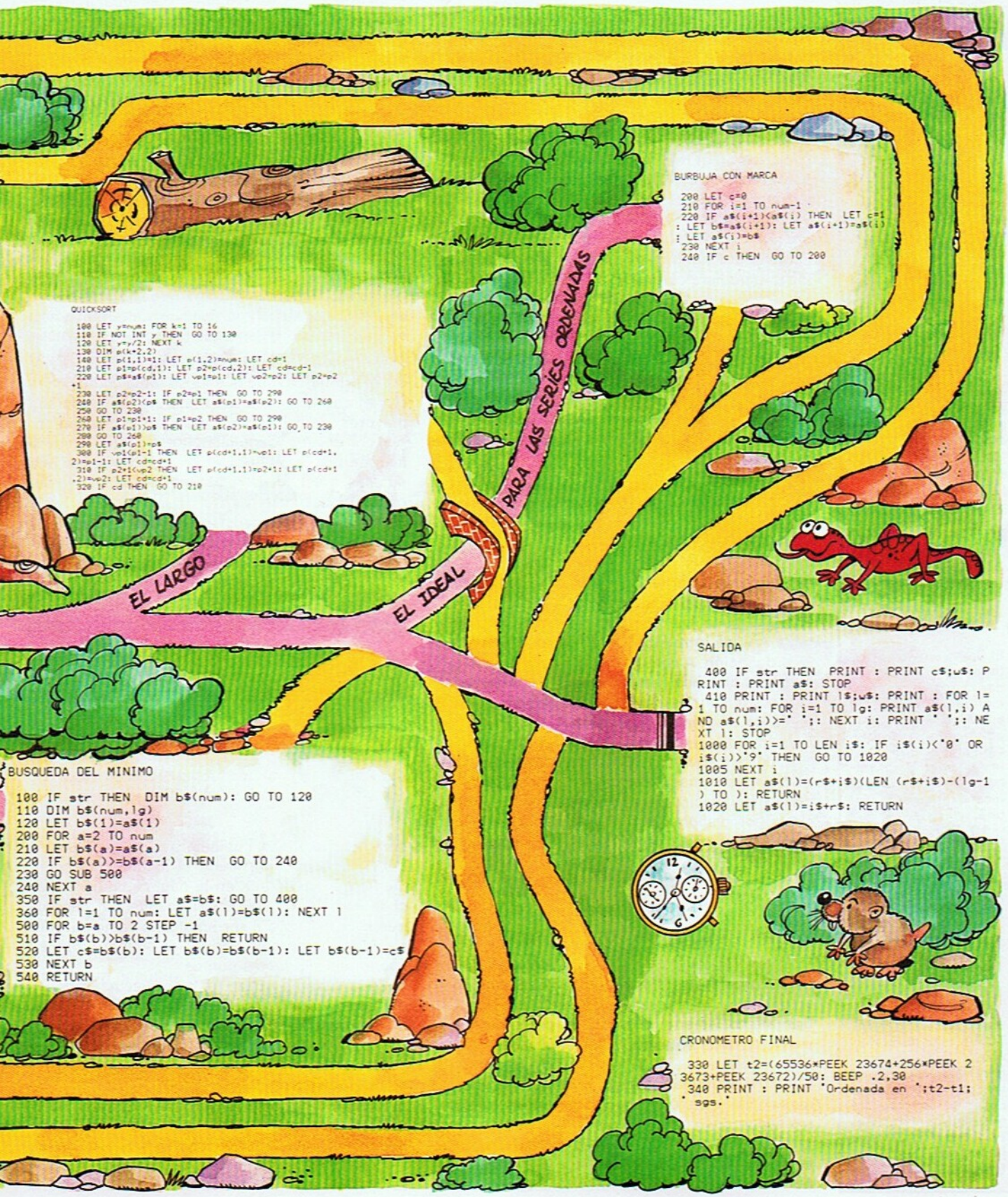
```
200 FOR i=1 TO num-1
210 LET m=i
220 FOR j=i+1 TO num: IF a$(j)<a$(
m) THEN  LET m=j
230 NEXT j
240 IF m<>i  THEN  LET b$=a$(i): L
ET a$(i)=a$(m): LET a$(m)=b$
250 NEXT i
```

BINARIO

```
200 FOR i=1 TO num-1
210 FOR j=i+1 TO num
220 IF a$(i)>a$(j) THEN  LET b$=a$
(i): LET a$(i)=a$(j): LET a$(j)=b$
230 NEXT j: NEXT i
```

CAMBIO DE TABLA

```
100 IF str THEN  DIM b$(num): GO TO 120
110 DIM b$(num,lg)
120 LET p=1: LET m$=CHR$ 255
200 LET e=1: LET i=1
210 IF i>=num-1 THEN  GO TO 240
220 IF a$(e)<a$(i+1) THEN  LET i=i+1: G
O TO 210
230 LET e=i+1: LET i=i+1: GO TO 210
240 LET b$(p)=a$(e): LET a$(e)=m$: LET
p=p+1
250 IF p<=num THEN  GO TO 200
350 IF str THEN  LET a$=b$: GO TO 400
360 FOR l=1 TO num: LET a$(l)=b$(l): NE
XT l
```



QUICKSORT

```
100 LET y=num: FOR k=1 TO 16
110 IF NOT INT y THEN  GO TO 130
120 LET y=y/2: NEXT k
130 DIM p(k+2,2)
140 LET p(1,1)=1: LET p(1,2)=num: LET cd=1
210 LET p1=p(cd,1): LET p2=p(cd,2): LET cd=cd-1
220 LET p$=a$(p1): LET vp1=p1: LET vp2=p2: LET p2=p2
+1
230 LET p2=p2-1: IF p2=p1 THEN  GO TO 290
240 IF a$(p2)>p$ THEN  LET a$(p1)=a$(p2): GO TO 260
250 GO TO 230
260 LET p1=p1+1: IF p1=p2 THEN  GO TO 290
270 IF a$(p1)>p$ THEN  LET a$(p2)=a$(p1): GO TO 230
280 GO TO 260
290 LET a$(p1)=p$
300 IF vp1<p1-1 THEN  LET p(cd+1,1)=vp1: LET p(cd+1,
2)=p1-1: LET cd=cd+1
310 IF p2+1<vp2 THEN  LET p(cd+1,1)=p2+1: LET p(cd+1
,2)=vp2: LET cd=cd+1
320 IF cd THEN  GO TO 210
```

BURBUJA CON MARCA

```
200 LET c=0
210 FOR i=1 TO num-1
220 IF a$(i+1)<a$(i) THEN  LET c=1
: LET b$=a$(i+1): LET a$(i+1)=a$(i)
: LET a$(i)=b$
230 NEXT i
240 IF c THEN  GO TO 200
```

SALIDA

```
400 IF str THEN  PRINT : PRINT c$;u$: P
RINT : PRINT a$: STOP
410 PRINT : PRINT l$;u$: PRINT : FOR l=
1 TO num: FOR i=1 TO lg: PRINT a$(l,i) A
ND a$(l,i)>=' ';: NEXT i: PRINT ' ';: NE
XT l: STOP
1000 FOR i=1 TO LEN i$: IF i$(i)<'0' OR
i$(i)>'9' THEN  GO TO 1020
1005 NEXT i
1010 LET a$(l)=(r$+i$)(LEN (r$+i$)-(lg-1
) TO ): RETURN
1020 LET a$(l)=i$+r$: RETURN
```

BUSQUEDA DEL MINIMO

```
100 IF str THEN  DIM b$(num): GO TO 120
110 DIM b$(num,lg)
120 LET b$(1)=a$(1)
200 FOR a=2 TO num
210 LET b$(a)=a$(a)
220 IF b$(a)>=b$(a-1) THEN  GO TO 240
230 GO SUB 500
240 NEXT a
350 IF str THEN  LET a$=b$: GO TO 400
360 FOR l=1 TO num: LET a$(l)=b$(l): NEXT l
500 FOR b=a TO 2 STEP -1
510 IF b$(b)>b$(b-1) THEN  RETURN
520 LET c$=b$(b): LET b$(b)=b$(b-1): LET b$(b-1)=c$
530 NEXT b
540 RETURN
```

CRONOMETRO FINAL

```
330 LET t2=(65536*PEEK 23674+256*PEEK 2
3673+PEEK 23672)/50: BEEP .2,30
340 PRINT : PRINT 'Ordenada en ';t2-t1;
' sgs.'
```

BITS

Ahí va un truco muy interesante: cómo hacer que las letras de un mensaje giren ante nuestros propios ojos.

```
10 CLS
20 FOR I=7 TO 0 STEP -1
30 POKE 23606, I
40 PRINT AT 0,0;"MENSAJE"
50 PAUSE 10
60 NEXT I
```

*

Para ciertas labores de cronometrado, nos vendrá muy bien disponer de una rutina que indicque en minutos el tiempo que la máquina lleva trabajando:

```
PRINT INT ((65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/3000)
```

*

El Spectrum no se distingue por su capacidad para la generación de sonidos, sin embargo, en algunas ocasiones podemos conseguir interesantes efectos. Para empezar a trabajar probemos la siguiente rutina:

```
10 FOR I=0 TO 9
20 FOR J=0 TO 39
30 BEEP .01,30
40 NEXT J
50 PAUSE 100
60 NEXT I
```

¿A que dan ganas de coger el teléfono?

*

En ocasiones el **RAM TOP** o tope de la memoria fijado por programación, puede ser tan bajo que no deje pasar ni un byte, ¡lo que se dice ni un byte! Para comprobarlo sólo hemos de introducir **CLEAR 23821**.

* BUSQUEDA DEL MINIMO

En todos los métodos vistos anteriormente, las comparaciones y los intercambios de lugar entre elementos se efectuaban sobre una única tabla. A continuación veremos dos sistemas que utilizan dos tablas para ejecutar el proceso de clasificación.
El primero de ellos, cada vez que realiza una pasada, extrae el menor de los elementos de la tabla A (tabla FUENTE con los datos desordenados) y lo coloca en la B (tabla DESTINO para los datos ya clasificados). Para recordar que éste se haya ordenado, introduce en su lugar de la tabla A un carácter sin significado, superior a cualquiera de los que pudiera contener la tabla. En el ejemplo, se ha escogido a tal fin un **CHR$ 255**, pero podría ser cualquier otro que desempeñara la misma labor.
Este método, que tan solo se presenta aquí como otra técnica a tener en cuenta, se revela bastante ineficaz, ya que los tiempos empleados en clasificar listas, parcialmente ordenadas o no, son los más altos entre todos los sistemas que estamos repasando.

* CAMBIO DE TABLA

Al igual que el anterior, utiliza dos tablas para efectuar la clasificación.
El SORT toma los dos primeros elementos de la tabla A y los introduce en la B, donde procede a su ordenación. A continuación, añade en B el tercero y lo clasifica respecto a los dos anteriores. El proceso continúa hasta haber clasificado en B todos los elementos de la tabla A.
Este método es especialmente recomendable cuando se tratan listas bastante ordenadas. Además, en este caso, la tabla A no ha sufrido ninguna modificación, lo que puede ser muy útil si queremos disponer simultáneamente en la memoria de la colección de datos clasificada y sin clasificar.

* SHELL-METZNER

Finalmente, he aquí dos métodos que, cuando el volumen de datos a tratar es elevado y se encuentran aleatoriamente distribuidos, proporcionan la mayor eficacia entre todos los analizados.
Más sofisticado, el SHELL-METZNER tiene un principio matemático algo más complicado: la idea consiste en realizar comparaciones entre datos separados en la lista por una distancia constante L=INT((2^P—1)/2); siendo P el exponente de la menor potencia de 2 que hace 2^P superior al número de elementos de la serie. Es decir, si por ejemplo tenemos 6 palabras a clasificar, P sería 3, ya que 2^3=8>6. Así pues, L es igual a 3 y por tanto comparamos la primera con la cuarta, la segunda con la quinta, la tercera con la sexta y las intercambiamos cuando sea necesario. A continuación, reducimos la distancia L a la mitad y repetimos el proceso hasta que la distancia sea nula, momento en que la lista estará ya clasificada.
Para utilizarlo, debemos pagar el precio de añadir algunas instrucciones más que en los métodos anteriores. Pero a cambio, el número de comparaciones se reduce sensiblemente, lo que proporciona un ahorro de tiempo considerable.

* EL QUICKSORT

El refinamiento clasificando llega con este método: el primer dato de la lista se toma como pivote con el que comparar los restantes. Si éstos son mayores, se colocan por debajo y si son menores, por encima. Seguidamente, fragmentamos la lista en dos, repitiendo el proceso con cada una de ellas, y así sucesivamente, mientras haya series para clasificar.
Se trata del clasificador que mayor cantidad de memoria necesita, pero su eficacia queda justificada en función del tiempo de proceso empleado.

LA FIGURA

Para la adopción de cualquiera de los métodos de SORT, debemos introducir el listado correspondiente que se detalla en la página siguiente, en cada uno de los claros del bosque. En caso de que el SORT vaya a ser empleado como subrutina de otro programa, tengamos siempre muy en cuenta los siguientes puntos:

* Cada sistema utiliza para su ejecución determinadas variables que pueden tener otra misión en el programa principal; de ser así, podemos optar por seguir dos caminos: o cambiamos las variables afectadas en el SORT, o hacemos lo propio en nuestro programa.

* Existen determinadas variables que los SORT utilizan como parámetros y que, por tanto, deben ser inicializadas antes de llegar a la subrutina; tales son: STR, que indica el tipo de serie a tratar (STR<>0, supone una cadena y STR=0 supone

un dimensionado alfanumérico). NUM, que es siempre el número de elementos de la serie y LG, que se emplea sólo en casos de dimensionado (STR=0) para indicar la segunda dimensión de la tabla (longitud de los elementos). Por supuesto, estas variables pueden ser cambiadas de nombre siempre y cuando el cambio sea realizado en toda la subrutina.

* Como soporte de la serie a clasificar se ha elegido la cadena o dimensionado A$, y como puente en los métodos que precisan de dos tablas, B$. Al igual que en los casos anteriores, estas variables pueden ser cambiadas de nombre, pero siempre respetando los cambios en toda la subrutina. Aunque en los ejemplos se utilizan series alfanuméricas, la adaptación de las subrutinas para que funcionen con tablas numéricas es extremadamente sencilla: prácticamente sólo es necesario escoger el sistema de clasificación de series de datos, eliminando el parámetro LG y los símbolos dólar **($)** que encontremos en la subrutina.

Como ya hemos dicho, la eficacia de cada SORT depende en su mayor parte de las características especiales de la serie a tratar (número de elementos y estado de desorden en que se encuentra). Para aquellos que quieran comprobar por sí mismos el funcionamiento de cada método, hemos dispuesto una interesante gira turística por el bosque de los SORT.

La etapa siempre comenzará en un mismo punto: la ENTRADA del bosque; una vez introducido este listado, podremos continuar nuestra andadura hasta el control de tiempo. Si deseamos conocer el tiempo en que el SORT va a clasificar la serie de datos, deberemos poner en marcha el cronómetro introduciendo las dos líneas que se señalan en este punto del recorrido. Siguiendo siempre los caminos rosas, podremos acceder al SORT que más nos convenga, y una vez introducido el correspondiente listado, abandonaremos el claro del bosque mediante el camino amarillo de salida.

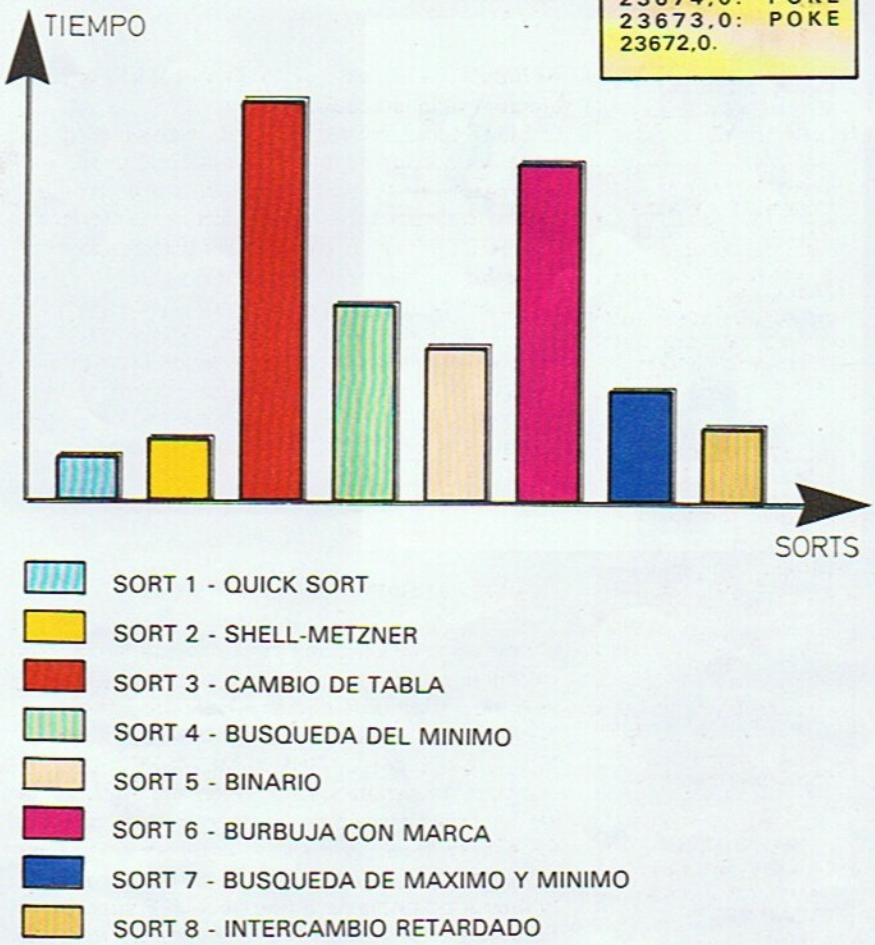
Este tipo de caminos nos llevarán siempre al control de fin de cronómetro, cuyas líneas deberemos introducir si deseamos conocer el tiempo de ejecución del SORT escogido. Finalmente, el programa SALIDA, marcará el fin de nuestro recorrido y gracias a él se mostrará en la pantalla la serie ordenada.

Al llegar a la salida del bosque, tendremos en la memoria de nuestro aparato un programa que podremos ejecutar para comprobar la eficacia del SORT escogido. Sólo queda aclarar las preguntas que el programa ENTRADA nos efectuará para realizar la inicialización del sub-programa de clasificación.

En primer lugar, en la ENTRADA se nos interrogará sobre el tipo de serie a clasificar: Cadena de caracteres (cadena) o cadena de datos (dimensionado). Si se escoge el primer punto, el programa nos facilitará la introducción de la cadena a clasificar y comenzará su desenfrenada carrera por el itinerario del bosque que hayamos escogido. Cuando se selecciona el punto segundo (cadena de datos), el programa, además de interrogar sobre dos cuestiones bien necesarias: número de datos y longitud de los mismos, presentará un nuevo menú de tres opciones, bajo el título de «Lógica». Si se selecciona el tercer punto (lógica A.S.C.I.I.), el programa elegirá como criterio de clasificación de la serie, la posición de los caracteres en el código A.S.C.I.I.

Si se selecciona la lógica numérica (punto segundo), los datos de la serie, aunque introducidos como cadenas, serán numéricos y por tanto serán clasificados como tales. Si por el contrario, el punto escogido es el primero (lógica mixta), los datos que componen la serie podrán ser tanto numéricos como textos, por lo que se realizará la clasificación pertinente para cada uno de los casos: los numéricos como tales y los textos como cadenas de caracteres.

Gráfico de tiempos medios de los diversos SORTs, tomados en la clasificación de series aleatorias de 100 elementos.



BITS

Es curioso lo que ocurre cuando llevamos nuestro ordenador a situaciones extremas, ¡hasta le bailan los caracteres! Probemos por ejemplo a forzar el **RAM TOP** a su punto más alto, es decir, **CLEAR 65535** en el modelo de 48 K. o **CLEAR 32767** en el de 16 K. Si a continuación escribimos los caracteres gráficos entre la L y la U, los veremos bailar al son de cualquier tecla.

*

Cuando queramos poner a cero el cronómetro, sólo tendremos que ejecutar: **POKE 23674,0: POKE 23673,0: POKE 23672,0.**

MODULO LUNAR

TRANSCURRE el año 2020; la conquista del espacio es una realidad y decenas de estaciones espaciales circunvalan la Tierra. Hace un decenio se construyó la primera base lunar de utilización conjunta internacional, LUNA-1.

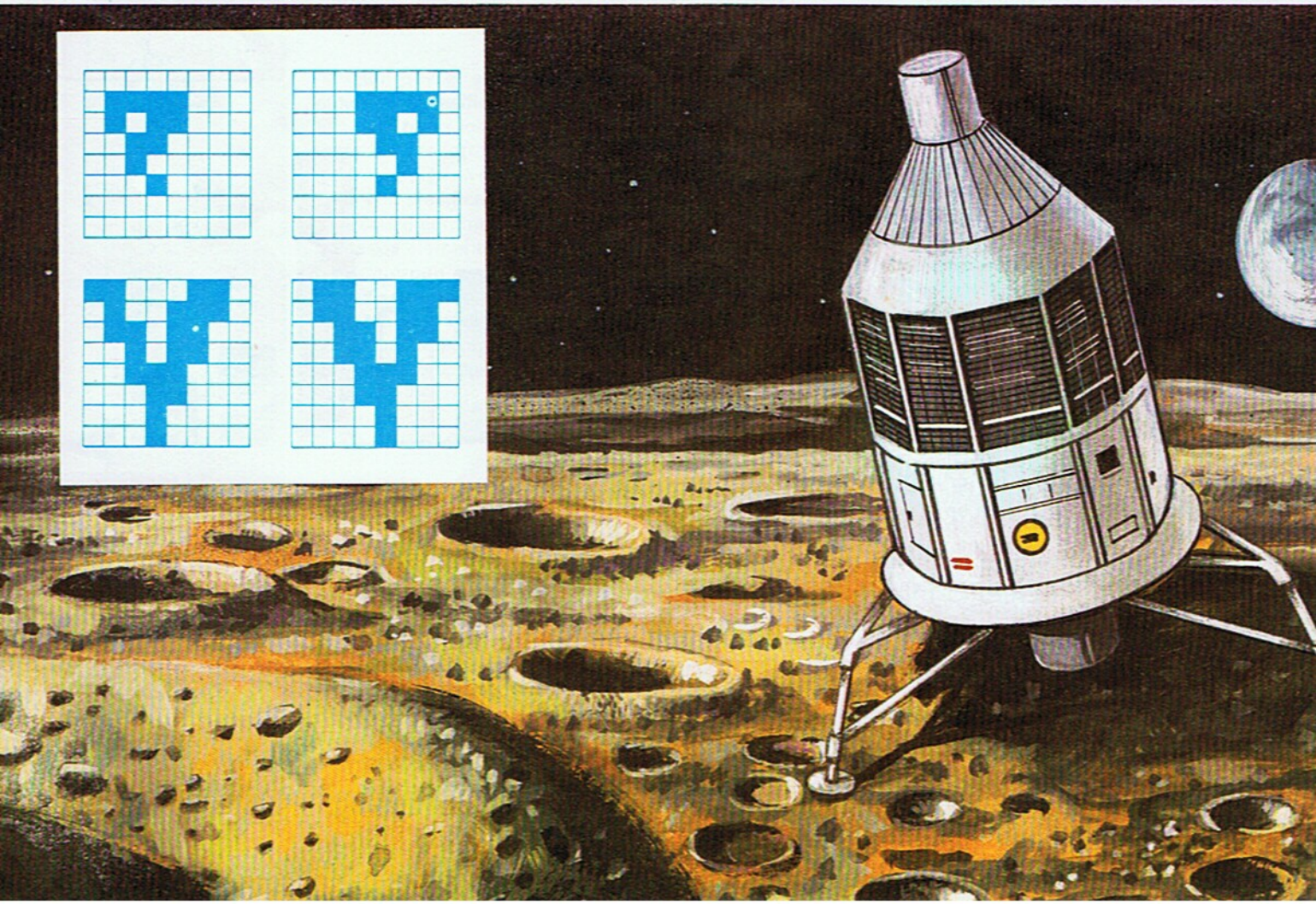
La función de LUNA-1 es purificar el plutonio que se extrae de las minas colindantes. Dicho mineral ha de ser enviado a otra base para su posterior proceso de elaboración; este transporte se efectúa a bordo de un módulo que tiene que sobrevolar las afiladas aristas de las cordilleras lunares.

Nosotros somos los encargados de pilotar el módulo lunar y posarnos, lo más suavemente posible, en la plataforma de destino, evitando destrozar la nave contra las montañas. Para el control del módulo disponemos tan solo de dos mandos: uno de ellos la hace ascender en vertical y el otro, descender en diagonal izquierda. Hay que tener en cuenta que la caída inercial de la nave cuando no le brindamos empuje, es en diagonal derecha.

Este tipo de desplazamientos de descenso, un tanto extraños, pueden atribuirse a poderosísimas razones, tales como el desplazamiento de las cargas de plutonio por el interior del módulo, alteraciones en los sistemas giroscópicos o perturbaciones de campos magnéticos. Ahora bien, lo cierto y verdad, es que obedecen a un único y transparente objetivo: complicar la tarea del alunizaje.

Así pues, los controles de la nave responden a las siguientes teclas: **P** => ARRIBA y **L** => ABAJO/IZQUIERDA. Al comienzo de cada intento de alunizaje, seremos interrogados acerca del nivel

Gráficos para los fogonazos de propulsión.



de dificultad con que deseamos operar (de 0 a 8). De este parámetro dependerá lo escarpado del perfil lunar que a continuación se presente.

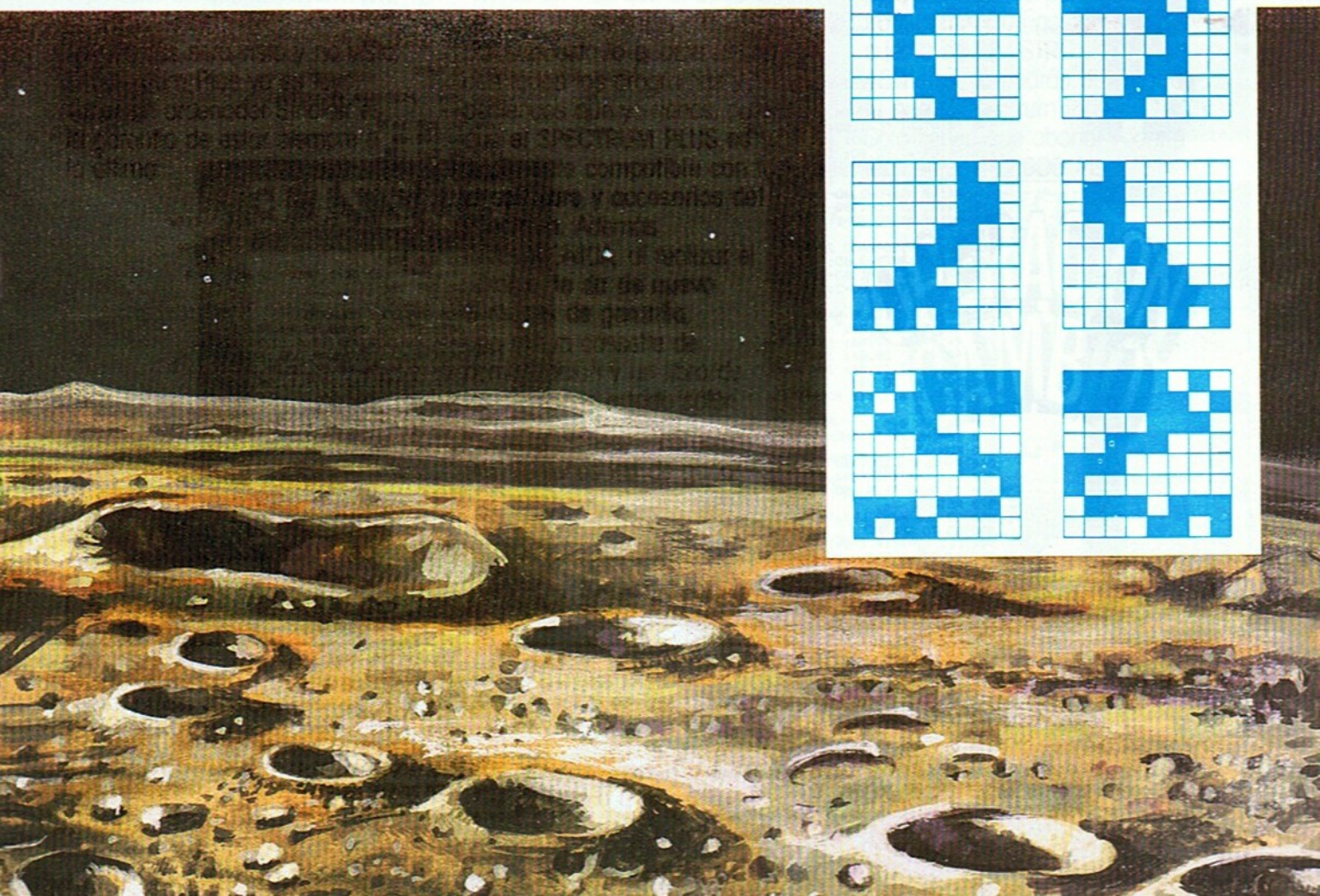
La arquitectura del programa es completamente estructurada. El bloque de programa (PROGRAMA PRINCIPAL) que controla todas y cada una de las situaciones (SUBRUTINAS) de la nave es muy simple y estricto. Como punto a destacar en el programa, podemos hablar de la animación conseguida mediante superposición de gráficos definidos.

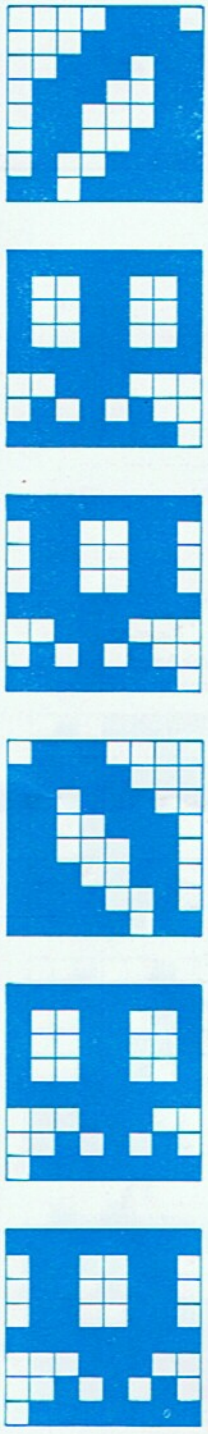
El programa tiene definidos dieciséis gráficos de usuario, de los cuales los ubicados en los caracteres CD y QR son los que simulan la rotación del anillo central de nuestro módulo lunar, y los definidos en los caracteres GH e IJ asemejan la amortiguación de las plataformas de alunizaje. El resto de los gráficos de usuario definen: zona superior de las plataformas de despegue y alunizaje (gráficos en EF), zona superior de la nave (gráficos en AB), llamas de propulsión (gráficos en KL y MN).

Aunque ya es costumbre, no está de más advertir, que en el listado, por motivos de comodidad a la hora de la introducción, los gráficos definidos han sido sustituidos por las letras subrayadas de las teclas mediante las cuales se obtienen. Así por ejemplo, una A deberá ser introducida como un carácter gráfico de la A, es decir: **GRAPHICS A GRAPHICS.**

Para la adopción del programa procederemos por el sistema habitual: **SAVE "MODULO"** o bien **SAVE "MODULO" LINE 10** (si deseamos que el programa se autoejecute al cargarlo).

Gráficos definidos para las bases de despegue y alunizaje.





Gráficos para la animación del módulo lunar.

```
 10 REM ************************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM ************************
 40 POKE 23658,8
 50 GO SUB 1910: GO TO 1570
 60 LET TO=0: LET FD=20: LET FD1=20: LET SW=0
 70 REM POSIC. ALEAT. BASES
 80 LET BD=INT (RND*13)+1
 90 LET BD1=BD
100 LET BA=INT (RND*14)+16
110 LET BA1=BA
120 RETURN
130 REM POSIC. B. DESPEGUE.
140 PRINT AT 21,BD; INK 5;"IJ"
150 RETURN
160 REM POSIC. B.ALUNIZ.
170 PRINT AT 20,BA; INK 5;"EF"
180 PRINT AT 21,BA; INK 5;"GH"
190 RETURN
200 REM DIBUJO PAISAJE.
210 LET XP=(BD*8)-3
220 LET YP=0
230 PLOT XP,YP
240 LET XP1=XP-((-5*ND)+47)
250 LET YP1=INT (RND*(50+10*ND))+1
260 IF XP1<=0 THEN  GO TO 310
270 DRAW  INK 4;XP1-XP,YP1-YP
280 LET XP=XP1
290 LET YP=YP1
300 GO TO 240
310 IF XP1=0 THEN  GO TO 320
320 DRAW  INK 4;-XP,INT (RND*130)-YP
330 REM
340 LET XP=((BD+1)*8)+10
350 LET YP=0
360 PLOT XP,YP
370 LET XP1=XP+((-5*ND)+47)
380 LET YP1=INT (RND*(50+ND*10))+1
390 IF XP1>BA*8-3 THEN  GO TO 440
400 DRAW  INK 4;XP1-XP,YP1-YP
410 LET XP=XP1
420 LET YP=YP1
430 GO TO 370
440 DRAW  INK 4;(BA*8-3)-XP,-YP
450 REM
460 LET XP=((BA+1)*8)+10
470 LET YP=0
480 PLOT XP,YP
490 LET XP1=XP+((-5*ND)+47)
500 LET YP1=INT (RND*(50+ND*10))+1
510 IF XP1>=255 THEN  GO TO 560
520 DRAW  INK 4;XP1-XP,YP1-YP
530 LET XP=XP1
540 LET YP=YP1
550 GO TO 490
560 DRAW  INK 4;255-XP,134-YP
570 RETURN
580 REM ROTAC.&MOVIM.NAVE
590 PRINT AT FD-1,BD; INK 6;"AB"
600 PRINT AT FD,BD; INK 6;"CD"
610 PRINT AT FD,BD; INK 6;"QR"
620 RETURN
630 REM SBR.BORRADO 1 -NAVE
640 PRINT AT FD-2,BD-1;"  "
650 PRINT AT FD-1,BD-1;" "
660 PRINT AT FD,BD-1;" "
670 RETURN
680 REM SBR.BORRADO 2- NAVE
690 PRINT AT FD-2,BD;"  "
700 PRINT AT FD-1,BD+2;" "
710 PRINT AT FD,BD+2;" "
720 RETURN
730 REM ATERRIZAJE NAVE
740 PRINT AT 18,BA;"  "
750 PRINT AT 19,BA; INK 5;"AB"
760 PRINT AT 20,BA; INK 5;"CD"
770 PRINT AT 21,BA; INK 6;"IJ"
780 FOR N=1 TO 200
790 BORDER 6:::::: BORDER 5:::::: BORDER 4::::::
800 NEXT N
810 BORDER 0
820 IF INKEY$="" THEN  GO TO 820
830 RUN 1570
840 REM SBR. TORTAZO
850 LET C=ATTR (FD+1,BD+1)
860 IF C<=5 THEN  GO TO 880
870 RETURN
880 GO SUB 640: LET TO=1
890 GO TO 950
900 LET C1=ATTR (FD+1,BD)
910 IF C1<=5 THEN  GO TO 930
920 RETURN
930 GO SUB 690: LET TO=1
940 REM  EXPLOSION
950 DIM D(4): LET FD1=FD: LET BD1=BD
960 LET FD2=FD-1: LET BD2=BD
970 LET FD3=FD-1: LET BD3=BD+1
980 LET FD4=FD: LET BD4=BD+1
990 PRINT AT FD1,BD1;"  ": LET D(1)=1
1000 PRINT AT FD2,BD2;"  ": LET D(2)=1
1010 PRINT AT FD3,BD3;"  ": LET D(3)=1
1020 PRINT AT FD4,BD4;"  ": LET D(4)=1
1030 IF FD1>20 OR BD1<1 THEN  LET D(1)=0
1040 IF FD2<1 OR BD2<1 THEN  LET D(2)=0
1050 IF FD3<1 OR BD2>30 THEN  LET D(3)=0
1060 IF FD4>20 OR BD4>30 THEN  LET D(4)=0
1070 IF D(1)=0 OR D(2)=0 OR D(3)=0 OR D(4)=0 THEN  BE
EP .5,30: BEEP .5,10: GO TO 820
1080 IF D(1)=1 THEN  BEEP .05,20: LET FD1=FD1+1: LET
BD1=BD1-1: PRINT AT FD1,BD1; INK 6;"C": GO TO 1090
1090 IF D(2)=1 THEN  BEEP .05,20: LET FD2=FD2-1: LET
BD2=BD2-1: PRINT AT FD2,BD2; INK 6;"A": GO TO 1100
1100 IF D(3)=1 THEN  BEEP .05,20: LET FD3=FD3-1: LET
BD3=BD3+1: PRINT AT FD3,BD3; INK 6;"B": PAUSE 2: GO T
O 1110
1110 IF D(4)=1 THEN  BEEP .05,20: LET FD4=FD4+1: LET
BD4=BD4+1: PRINT AT FD4,BD4; INK 6;"D": PAUSE 2: GO S
UB 1120: GO TO 1030
1120 PRINT AT FD1,BD1;" "
1130 PRINT AT FD2,BD2;" "
1140 PRINT AT FD3,BD3;" "
1150 PRINT AT FD4,BD4;" "
1160 RETURN
1170 REM ARRIBA NAVE
1180 IF FD=1 THEN  LET FD=2
1190 LET FD=FD-1
1200 GO SUB 590
1210 PRINT AT FD+1,BD; INK 6;"KL";AT FD+1,BD; INK 2;"
MN"
1220 IF FD<=18 THEN  PRINT AT FD+2,BD;"  "
1230 IF FD=19 OR FD=18 THEN  GO TO 1250
1240 RETURN
1250 PRINT AT FD1,BD1; INK 2;"EF"
1260 PRINT AT FD1+1,BD1; INK 2;"GH"
1270 RETURN
1280 REM ABAJO IZQUIERDA
1290 LET FD=FD+1
1300 LET BD=BD-1: IF TO=1 THEN  RETURN
1310 IF BD<=0 THEN  LET BD=0
1320 GO SUB 690: GO SUB 590
1330 IF BD=BA AND FD=19 THEN  GO SUB 690: GO SUB 740
1340 GO SUB 900
1350 GO SUB 690
1360 RETURN
1370 REM ABAJO DERECHA
1380 LET FD=FD+1
1390 LET BD=BD+1: IF TO=1 THEN  RETURN
1400 IF BD=31 THEN  LET BD=30
1410 GO SUB 640: GO SUB 590
1420 IF BD=BA AND FD=19 THEN  GO SUB 640: GO SUB 740
1430 GO SUB 850
1440 GO SUB 640
1450 RETURN
1460 REM NIVEL DIFICULTAD
1470 BORDER 0: PAPER 0: INK 9
1480 CLS
1490 PRINT  PAPER 6; BRIGHT 1;AT 13,3;"  NIVEL DE DIF
ICULTAD  "
1500 PRINT AT 15,11;"0 - 8"
1510 LET N$=INKEY$
1520 IF CODE N$<48 OR CODE N$>56 THEN  GO TO 1510
1530 BEEP .2,20
1540 LET ND=VAL N$
1550 CLS
1560 RETURN
1570 REM
1580 REM   PROGRAMA PRINCIPAL.
1590 REM
1600 GO SUB 1470
1610 GO SUB 60
1620 GO SUB 140
1630 GO SUB 170
1640 GO SUB 210
1650 LET K$=INKEY$
1660 GO SUB 590
1670 IF K$="P" THEN  LET SW=1: GO SUB 1180
1680 GO SUB 590
1690 IF K$="L" AND SW=1 THEN  GO SUB 1290
1700 GO SUB 590
1710 IF K$="" AND SW=1 THEN  GO SUB 1380
1720 GO SUB 590
1730 GO TO 1650
1740 REM GRAFICOS USUARIO
1750 DATA 95,166,95,48,32,48,24,12
1760 DATA 250,101,250,12,4,12,24,48
1770 DATA 4,6,6,12,24,48,248,168
1780 DATA 32,96,96,48,24,12,31,21
1790 DATA 95,191,112,28,7,1,238,176
1800 DATA 250,253,14,56,224,128,119,13
1810 DATA 14,31,59,115,99,103,79,208
1820 DATA 112,248,220,206,198,230,242,11
1830 DATA 255,153,153,153,255,56,84,254
1840 DATA 255,153,153,153,255,28,42,127
1850 DATA 0,120,88,48,32,16,0,0
1860 DATA 0,30,26,12,4,8,0,0
1870 DATA 0,102,108,44,56,24,16,16
1880 DATA 0,114,54,52,28,24,8,8
1890 DATA 255,102,102,102,255,56,84,254
1900 DATA 255,102,102,102,255,28,20,127
1910 LET G$="EFGHIJABCDKLMNQR"
1920 RESTORE 1750
1930 GO SUB 2030
1940 FOR M=1 TO LEN G$
1950 FOR N=0 TO 7
1960 READ D
1970 POKE USR G$(M)+N,D
1980 BEEP .01,30
1990 NEXT N
2000 BEEP .1,10
2010 NEXT M
2020 RETURN
2030 PRINT  FLASH 1; INK 1; PAPER 6;AT 10,3;"
2040 PRINT  FLASH 1; PAPER 6; INK 1;AT 11,3;"  "; PAP
ER 1; INK 6;" UN MOMENTO POR FAVOR "; INK 1; PAPER 6;
"
2050 PRINT  FLASH 1; INK 1; PAPER 6;AT 12,3;"
2060 PRINT  PAPER 2; INK 7;AT 14,6;" ~GRAFICOS USUARI
O~ "
2070 RETURN
```

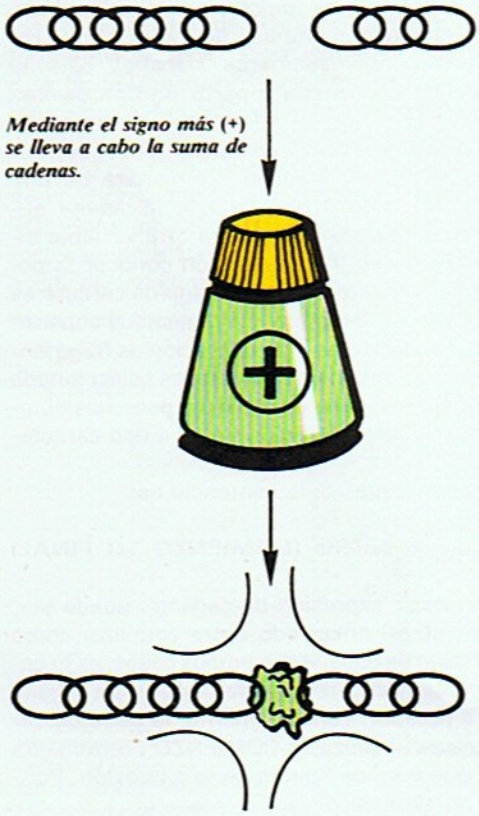
ROMPIENDO CADENAS

Ya hemos visto que el Spectrum trata con dos tipos de variables muy diferentes: las numéricas y las alfanuméricas o de cadena.

Las variables numéricas contienen únicamente valores numéricos, pudiendo estar su nombre formado por cualquier combinación de caracteres alfabéticos o numéricos, siendo necesariamente alfabético el primero de ellos. Entre ellas, podemos establecer comparaciones y, por supuesto, realizar todo tipo de cálculos matemáticos.

Los nombres de las variables alfanuméricas tienen reglas más estrictas de formación: deben estar compuestos por un solo carácter seguido del símbolo dólar ($). Estas variables se denominan también de cadena o *strings,* debido a que están formadas por una cadena o serie de caracteres cualquiera, que pueden incluir combinaciones de alfabéticos, numéricos y símbolos especiales.

Entre cadenas se pueden establecer comparaciones y operaciones especiales de unión («suma») y fragmentación. Estas son las que a continuación estudiaremos.



Mediante el signo más (+) se lleva a cabo la suma de cadenas.

Las variables de cadena contienen todo tipo de literales, con los que se pueden realizar operaciones de unión y fragmentación.

LA UNION HACE LA FUERZA

La operación más sencilla que podemos realizar con cadenas es la suma. Cuando hablamos de suma de cadenas, nos referimos a crear una cadena de caracteres única, que contenga los caracteres de dos (o más) cadenas anteriormente definidas. Esta operación de suma no es conmutativa como sucede con la numérica, es decir, no es lo mismo sumar de la forma **A$+B$** que **B$+A$**, aunque, en ambos casos, la cadena resultado de la operación tiene como longitud la suma de las longitudes de las cadenas sumandos, siendo el número de sumandos ilimitado. Lo veremos más claro en un ejemplo:

```
10 REM - SUMA DE CADENAS
20 CLS
30 LET A$="Estamos aprendiendo  "
40 LET B$"a programar en BASIC."
50 PRINT  "A$=";A$"
60 PRINT  "B$=";B$"
70 LET C$=A$+B$
80 PRINT  "C$=";C$
```

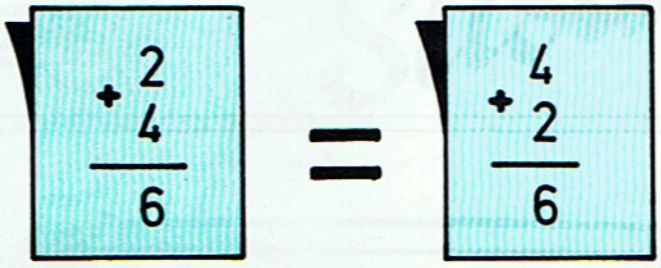
¡!

Si la cadena de sustitución es vacía (dos comillas consecutivas), se sustituyen todos los caracteres por espacios.

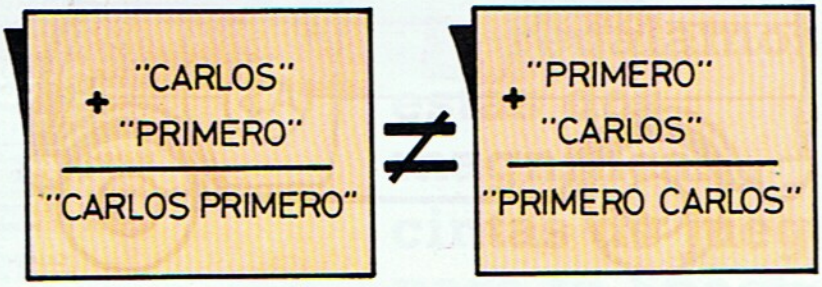
*

Cuando la cadena de sustitución es mayor que la cadena raíz, queda truncado el cambio de los caracteres que exceden la longitud de la parte a sustituir.

CONMUTATIVA:



NO CONMUTATIVA:



La operación «suma» (unión) realizada con cadenas no es conmutativa, es decir, el orden en que aparezcan los «sumandos» sí altera el resultado final.

Hemos obtenido una nueva cadena de caracteres (**C$**), resultado de la suma de otras dos (**A$** y **B$**), cuya longitud es la suma de las dos anteriores, sin perder los contenidos originales que, antes de realizar la operación, tenían las dos variables sumandos. De haber invertido el orden de los sumandos en la operación, habríamos obtenido un resultado no deseado en **C$**:

a programar en BASIC.Estamos aprendiendo

En cualquier caso, debemos recurrir a la suma de cadenas sólo cuando nos sea muy necesario. Así pues, en nuestro ejemplo podríamos haber evitado la operación, escribiendo el **PRINT** con punto y coma (;), para ahorrar así espacio de memoria y ganar en claridad del programa:

```
10 REM - SUMA DE CADENAS
20 CLS
30 LET A$="Estamos aprendiendo  "
40 LET B$="a programar en BASIC."
50 PRINT  "A$+B$=";A$;B$
```

En otras ocasiones, es más cómodo recurrir a la suma de cadenas, y desde luego, de ser así, no debemos privarnos del uso de esta herramienta para la programación.

DIVIDE Y VENCERAS

Sin duda, dentro de las operaciones que se pueden realizar con cadenas de caracteres, la fragmentación es la más importante de todas ellas. Esta operación, denominada en inglés *slicing* (se pronuncia aproximadamente *slaising),* permite obtener una subcadena a partir de una cadena dada.

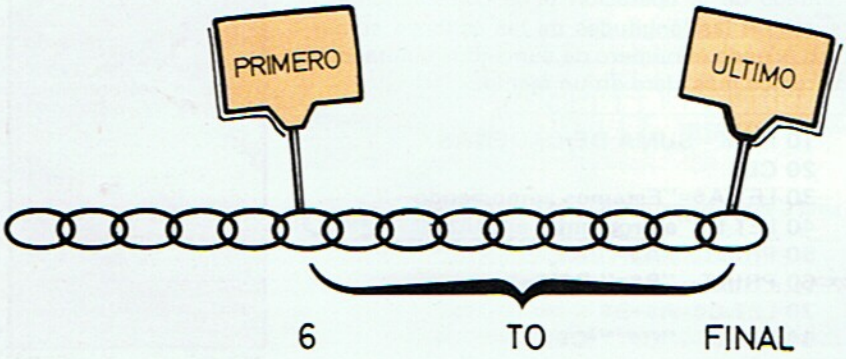
Por este procedimiento podemos conseguir, por ejemplo, el primer apellido del nombre de una persona contenido en una variable de cadena, pudiendo imprimirlo o asignarlo a otra variable diferente. Para ello, nos basta con conocer la posición relativa, dentro de la cadena de caracteres original, de la subcadena que deseamos separar. La única limitación de esta operación de fragmentación es que la serie de caracteres seleccionada debe ser consecutiva, es decir, no podemos obtener con una sola operación de este tipo caracteres no contiguos.

El formato general de la sentencia es:

expresión-de-cadena (COMIENZO TO FINAL)

El parámetro "expresión-de-cadena" puede ser, tanto un literal encerrado entre comillas, como una variable de cadena. En ambos casos, es la cadena alfanumérica de partida, a partir de la cual vamos a obtener, concretamente, la serie de caracteres que empieza en COMIENZO y termina en FINAL, que son los límites de la expresión. Pongamos un ejemplo:

Si no se especifica el parámetro FINAL, la extracción de una subcadena se realiza desde el primer carácter, indicado por COMIENZO, hasta el último de la cadena original.



PRINT "ROBERTO RODRIGUEZ GONZALEZ" (9 TO 17)

En nuestro caso, obtendremos en la pantalla el primer apellido del nombre: "RODRIGUEZ"; puesto que ésta es la subcadena de caracteres dentro de la dada como partida, formada por los caracteres noveno a decimoséptimo, como indicamos en los parámetros de COMIENZO y FINAL de **TO**.

Como podemos ver, la palabra BASIC **TO** (pronunciado *tu),* cuyo significado en inglés es HASTA, tiene una utilidad más, a parte de la que ya vimos en el control de bucles, cuando se utilizaba conjuntamente con **FOR, STEP** y **NEXT.**

Existe además la posibilidad de no emplear todos los parámetros en las instrucciones de *sciling.* Cuando no se especifica el COMIENZO, el Spectrum supone que la cadena resultado es la formada por todos los caracteres, a partir del primero, hasta el FINAL indicado por el segundo parámetro. Para obtener el nombre de pila de la cadena anterior, debemos escribir:

PRINT "ROBERTO RODRIGUEZ GONZALEZ" (TO 7)

De forma análoga, si omitimos el parámetro FINAL, el Spectrum supone la longitud total de la cadena por lo que, para extraer como subcadena el segundo apellido, debemos escribir:

PRINT "ROBERTO RODRIGUEZ GONZALEZ" (19 TO)

Existe, además, el caso concreto de que deseemos obtener un único carácter, el situado en determinada posición. Para ello, podemos recurrir a la fórmula general, especificando como parámetro de COMIENZO y FINAL el mismo número, o utilizar una versión simplificada de la sentencia, que consiste en colocar entre paréntesis el número de carácter que deseamos obtener (sin **TO**). Por ejemplo, para escribir en pantalla el carácter número 10 de la cadena (una **O**), podemos codificar, indistintamente:

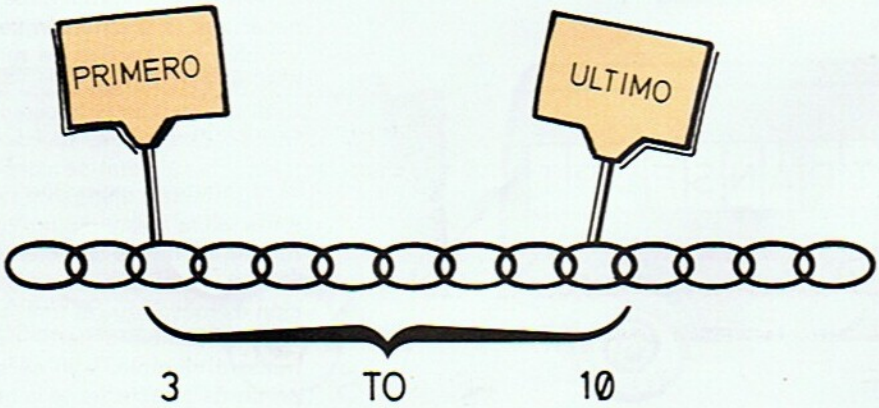
PRINT "ROBERTO RODRIGUEZ GONZALEZ" (10 TO 10)

Con la palabra BASIC **TO**, *conseguimos el «troceado» de cadenas o slicing.*

PRINT "ROBERTO RODRIGUEZ GONZALEZ" (10)

ALGUNAS ADVERTENCIAS

A la hora de especificar los parámetros de fragmentación, se deben tener en cuenta algunas re-



La función de troceado de cadenas utiliza dos parámetros, para indicar el primer y último carácter de la subcadena a obtener.

¡!

Si no se especifica parámetro de FINAL, el Spectrum supone la longitud total de la cadena, a partir del primer parámetro (COMIENZO).

*

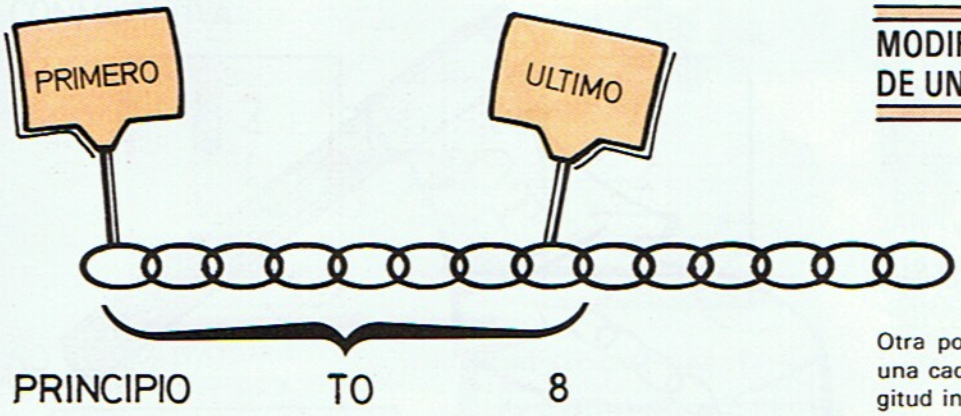
Para obtener un único carácter, el situado en determinada posición, basta con escribir entre paréntesis la posición relativa dentro de la cadena.

*

No se pueden especificar como parámetros posiciones de la cadena superiores a su longitud, ni valores negativos.

*

Si la sustitución se hace por una cadena con menos caracteres, todos los que faltan hasta completar la longitud total se consideran como espacios.



Al no especificar el parámetro COMIENZO para el slicing, la subcadena se toma desde el principio hasta el parámetro indicado como FINAL.

La serie de caracteres seleccionada en una fragmentación debe ser consecutiva, es decir, no podemos obtener con una sola operación de este tipo caracteres no contiguos.

El formato general de la sentencia de fragmentación es:

expresión-de-cadena (COMIENZO TO FINAL)

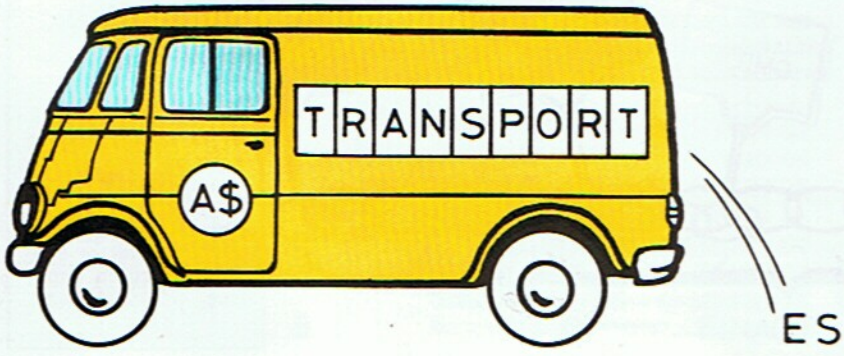
El parámetro "expresión-de-cadena" puede ser, tanto un literal encerrado entre comillas, como una variable de cadena.

glas. La primera de ellas es que no debemos incluir posiciones de la cadena superiores a su longitud, en evitación de errores.
Obtendremos un error del tipo **B, Integer out of range,** si especificamos, por ejemplo:

```
PRINT "ROBERTO RODRIGUEZ GONZALEZ"
  (25 TO 27)
```

Esto es debido a que la longitud total de la cadena es de 26 caracteres, de esta forma, pueden obtenerse los caracteres 25 y 26, pero nunca el 27 que se encuentra fuera de ella.
Otro de los detalles que debemos cuidar, es el de no utilizar valores negativos o cero para el COMIENZO y el FINAL de **TO;** caso de hacerlo, el programa se detendría emitiendo el mismo error, ya que los límites especificados no tienen ningún sentido.

Cuando un literal no «cabe» en una variable de cadena, se trunca por la derecha.



MODIFICACION DEL CONTENIDO DE UNA CADENA

Otra posibilidad interesante es la de modificar una cadena de caracteres, manteniendo su longitud inicial, y cambiando una serie de caracteres por otra en su contenido. Esto es lo mismo que ahorrarse el trabajo de fragmentar la cadena primero, para volverla a unir después por medio de la operación de suma con el contenido de la nueva subcadena. Lo vamos a ver más claro con un ejemplo, en el que sustituiremos el primer apellido por otro:

```
10 REM - MODIFICACION DE CADENA
20 CLS
30 LET X$="ROBERTO RODRIGUEZ GONZALEZ"
40 PRINT X$''
50 LET X$(9 TO 17)="REVUELTAS"
60 PRINT X$
```

Codificando de esta forma las instrucciones, nos evitaremos recurrir a sistemas más aparatosos, como el descrito en el siguiente párrafo, para obtener los mismos resultados:

```
10 REM - MODIFICACION DE CADENA
20 CLS
30 LET X$="ROBERTO RODRIGUEZ GONZALEZ"
40 PRINT X$''
50 LET X$=X$(TO 8)+"REVUELTAS"+X$(18 TO)
60 PRINT X$
```

De cualquiera de estas dos formas, cambiamos un primer apellido de nueve caracteres por otro de la misma longitud. Cabe preguntarnos qué sucedería si la sustitución se hace por un literal o variable con menos de nueve caracteres o con más de esta cifra.
Si la sustitución la hacemos por una cadena con menos caracteres, todos los que faltan hasta completar el total se consideran como blancos. Si, en nuestro ejemplo, la cadena de sustitución fuera vacía (dos comillas consecutivas), se sustituirían los nueve caracteres por espacios.
En caso contrario, cuando la cadena de sustitución es mayor que el total a sustituir, queda truncado el cambio a partir del último carácter. En nuestro ejemplo, sólo se sustituirían los nueve primeros caracteres de la nueva cadena, ignorán-

dose del diez en adelante, aunque no se emitiría ningún mensaje de error.

Existe, por último, una fórmula para reemplazar el contenido de una variable de cadena por el de otra, o por un literal entrecomillado, muy similar a la asignación directa de variables de cadena por medio de la sentencia **LET**, pero con algunas diferencias de matiz.

Este procedimiento nos permite obtener una variable de cadena de longitud fija, e independiente del tamaño del dato que coloquemos en ella, mediante las sentencias **INPUT** o **LET**. En otras palabras, podemos conseguir que, en una lista de nombres, por ejemplo, que aceptamos del teclado mediante instrucciones **INPUT**, todos ellos aparezcan impresos en la pantalla con el mismo número de caracteres, a pesar de haberse introducido con longitudes diferentes; recurriremos a las propiedades antes mencionadas, por las que de ser la cadena de sustitución más corta se rellena con blancos, y de ser más larga se trunca. Todo esto lo veremos más claro en el siguiente programa:

```
10 REM - LISTA DE NOMBRES
20 CLS:LET X$="********"
30 INPUT "Numero de nombres:";N
40 IF N<0 OR N>80 THEN GO TO 30
50 FOR I=1 TO N
60 INPUT N$
70 LET X$( )=N$
80 PRINT X$;
90 NEXT I
```

Este programa nos va a servir un poco de repaso de lo que ya sabemos sobre el BASIC. Por tanto, sigamos las explicaciones siguientes fijándonos en el listado.

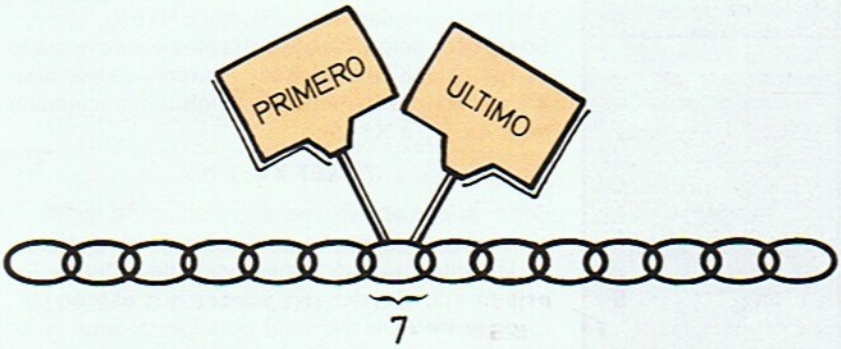
En la línea 20, primero borramos la pantalla por medio del comando **CLS** para, a continuación, asignar con la sentencia **LET** el valor de ocho asteriscos a la variable de cadena **X$**. Debemos observar que, como lo que realmente queremos hacer con **X$** es asignarle una longitud de ocho posiciones, lo mismo da que utilicemos asteriscos que cualquier otro carácter del teclado.

En la instrucción 30, hacemos un **INPUT** de la variable numérica **N**, que va a contener el número de nombres que queremos incluir en nuestra lista, sirviendo de tope a la variable **I** del bucle **FOR-NEXT**.

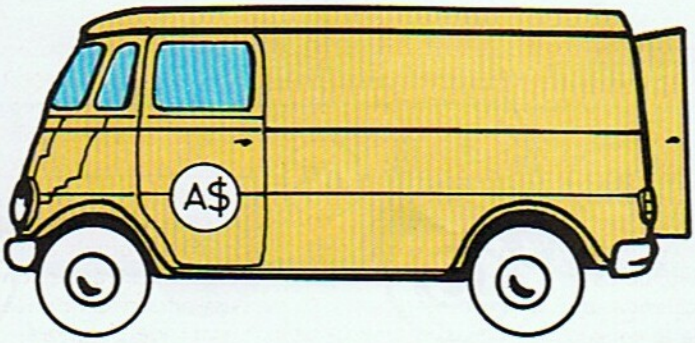
En la instrucción 40 hacemos una pequeña depuración del dato introducido en la línea anterior, verificando que no sea negativo (**<0**) ni mayor que un determinado número (**>80**), considerado ya como muy grande. Si en esta depuración se de-

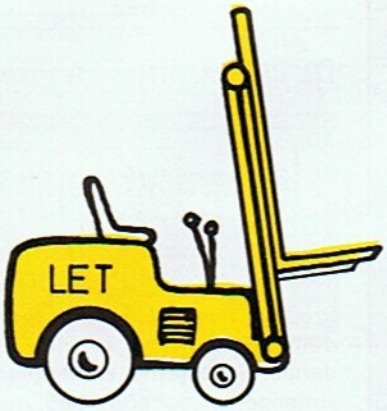
Cuando un literal no se ajusta a las dimensiones de la variable de cadena a que se pretende asignar, se completa con espacios en blanco por la derecha.

La forma abreviada de escoger un único carácter de una cadena es marcar su posición entre paréntesis.



Las sentencias **LET** *e* **INPUT** *se emplean para introducir los caracteres que componen las variables de cadena.*





Entre variables de cadena se pueden establecer comparaciones y operaciones especiales de unión y fragmentación.

*

La suma de cadenas permite crear una cadena de caracteres resultado, que contenga los caracteres de las cadenas sumando.

*

La suma de cadenas no es conmutativa, es decir, no es lo mismo sumar **A$+B$** que **B$+A$**, aunque, en ambos casos, la cadena resultado de la operación tiene como longitud la suma de las longitudes de las cadenas sumandos, siendo el número de sumandos ilimitado.

*

La fragmentación de cadenas de caracteres, permite extraer una subcadena de una cadena dada, conociendo la posición relativa de la subcadena que deseamos separar.

tecta algún error, el programa bifurca a la instrucción anterior, donde se vuelve a pedir el dato **N**.

En la línea 50 comienza el bucle de **N** pasadas, tantas como las especificadas en el **INPUT** de la instrucción 30.

En la instrucción 60 se aceptan, esta vez sin mensaje de petición, los nombres de la lista. Esta petición se repetirá, si nosotros no disponemos lo contrario, el número de veces especificado por **N**. Si deseamos interrumpir la ejecución del programa durante la introducción de los nombres, podemos hacerlo por dos procedimientos:

El primero consiste en salirnos de las comillas con la tecla de cursor a la derecha y teclear dos veces **DELETE**, seguido por **STOP** y **ENTER**. El segundo procedimiento, algo más sencillo, consiste en responder al **INPUT** con **EDIT (SHIFT + 1)**, gracias a lo cual desaparecerán las comillas, y pulsar seguidamente **STOP** y **ENTER**.

En la instrucción 70, encontramos el nuevo modo de reemplazo de una cadena caracteres por otra, **X$** por **N$** manteniendo la longitud fija asignada de principio a **X$** (8):

```
70 LET X$( )=N$
```

Como puede apreciarse, consiste en no codificar absolutamente ningún parámetro de los posibles, en la sentencia de fragmentación de cadenas. De esta forma, el ordenador supone que el reemplazo debe hacerse desde el principio al final de la cadena raíz, siguiendo la norma de relleno con espacios o truncamiento, dependiendo de la longitud de la serie de caracteres que debe sustituir a la variable original.

La instrucción 80 imprime el valor de **X$** ajustado a ocho posiciones fijas, utilizando el punto y coma (;), por lo que cada nuevo valor quedará inmediatamente a la derecha del anterior, dentro de la misma línea de la pantalla. De esta forma, obtendremos series de nombres de ocho caracteres, cuatro en cada línea.

Por último, la instrucción 90 cierra el bucle **FOR-NEXT** de la variable **I**, que tiene como límite **N** (número de entradas que se define al comienzo del programa).

DE LA TEORIA A LA PRACTICA

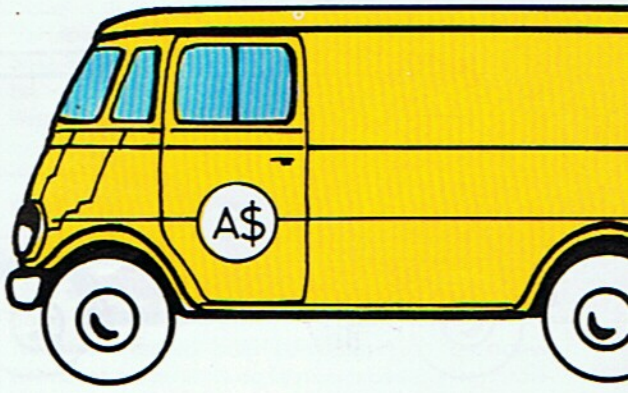
Proponemos ahora un breve programa, ejemplo de la utilidad de la fragmentación *(slicing)* de cadenas. Su objetivo es facilitarnos un calendario «inteligente» del año 1985, que controla la coherencia de la fecha introducida y la transforma en el formato convencional de día de la semana, día del mes, nombre del mes y año (en nuestro caso, siempre 1985):

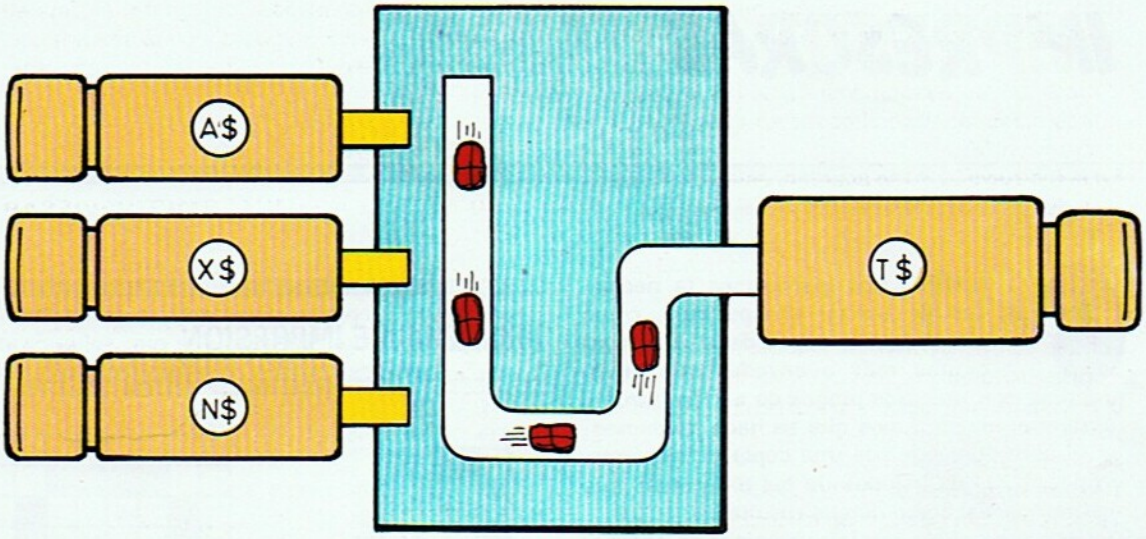
```
 10 REM - CALENDARIO 1985
 20 CLS
 30 LET D$=''LUNMARMIEJUEVIESAB
DOM''
 40 LET M$=''ENEFEBMARABRMAYJUN
JULAGOSEPOCTNOVDIC''
 50 LET L$="312831303130313130313031"
 60 INPUT ''Fecha:'';D,M:IF M<1 OR M>12
THEN GO TO 60
 70 IF D<1 OR D>VAL L$(2*M-1 TO 2*M)
THEN GO TO 60
 80 LET J=0
 90 FOR I=1 TO M-1
100 LET J=J+VAL L$(2*I-1 TO 2*I)
110 NEXT I
120 LET J=J+D:LET J=J-7*INT(J/7)+1
130 LET X$=''  '':IF D<10 THEN LET
X$=''  ''
140 PRINT D$(3*J-2 TO 3*J);X$;D;
''  '';M$(3*M-2 TO 3*M);''  1985'';
150 GO TO 60
```

En las líneas 30 a 50, se asignan los valores a las cadenas de caracteres que nos sirven para depurar y componer la fecha. **D$**, contiene los días de la semana; **M$**, los nombres de los meses del año, y **L$** el número de días de cada mes. En la línea 60 se aceptan los datos por medio de un **INPUT** de dos variables, **D** y **M**. La primera contendrá el día del mes y la segunda el número del mes (1 a 12); cualquier otro número que introduzcamos no será aceptado.

En la línea 70 se completa la depuración de la fecha, verificando que el número de día no sea negativo ni superior al número de días que tiene su mes correspondiente. Para ello recurrimos a una

Siempre que podamos imprimir cadenas unidas por punto y coma (;), debemos evitar la creación de nuevas cadenas como suma de anteriores.





Una nueva cadena se puede obtener como suma de dos o más cadenas anteriormente definidas.

fórmula, por la que calculamos con qué tramo de la cadena de caracteres **L$** debemos comparar, para que **D** no exceda ese límite.

Nos hemos permitido la licencia de incluir una sentencia, aún no comentada, por ser necesaria en el programa. Esta sentencia es **VAL**, y diremos de ella, solamente, que permite obtener el valor numérico de una variable de cadena, o parte de ella, como es nuestro caso. En las próximas páginas dedicadas al BASIC, entraremos más en profundidad en el estudio de esta función.

El siguiente cálculo a realizar, para obtener el día de la semana que corresponde a una fecha, es traducir ésta de la forma día y mes, al de fecha «juliana». Este sistema consiste en considerar un oriegen de fechas, en nuestro caso el día primero de enero del año, y a partir de él calcular el número de días transcurridos hasta la fecha introducida en cuestión. De esta manera, conociendo el día de la semana que fue el 1 de enero, es fácil calcular el que corresponde a un determinado número de días más tarde.

En la instrucción 80, ponemos a cero la variable **J**, que contendrá la fecha en formato juliano. A continuación, entramos en un bucle **FOR-NEXT**, en el que sumamos a **J** el total de días de los meses transcurridos hasta el corriente de la fecha introducida, para después sumar los días del mes en curso. De esta forma, reducimos cualquier fecha del año a un número de 1 a 365.

En la línea 120, se suman los días del mes corriente a **J**, y se calcula el día de la semana, cuyo literal obtenemos de la variable **D$**.

En la línea 130, recurrimos a un sistema de ajuste de las fechas para su impresión en la pantalla, puesto que las fechas cuyo día es menor de 10 ocupan un lugar menos. La variable **X$**, sirve para este ajuste.

Por último, en la línea 140, imprimimos el literal del día de la semana, el ajuste de espacios **X$**, el literal del número del mes y el literal fijo del año en curso, para volver al **INPUT** inicial en la siguiente instrucción.

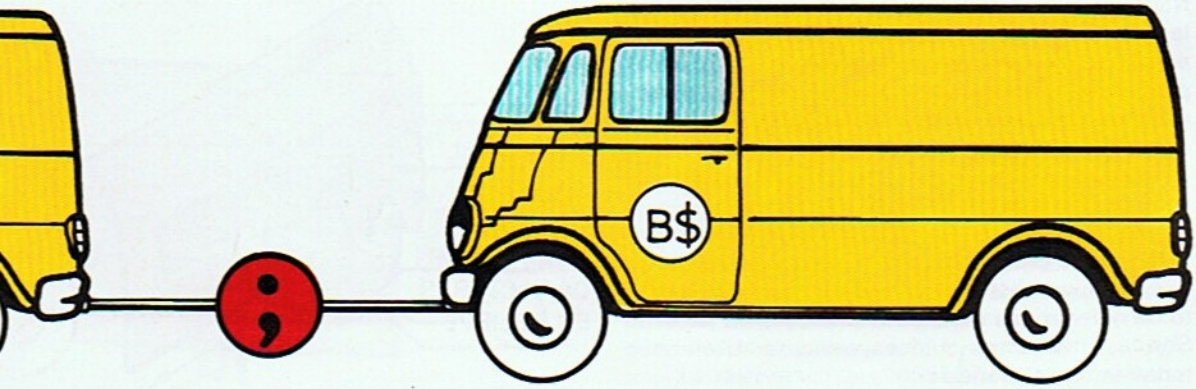
Para interrumpir el programa, basta con pulsar **STOP** y **ENTER** durante la ejecución de un **INPUT**.

¡!

Si no se especifica parámetro de COMIENZO, el Spectrum supone que la cadena resultado es la formada por todos los caracteres, a partir del primero, hasta el FINAL indicado por el segundo parámetro.

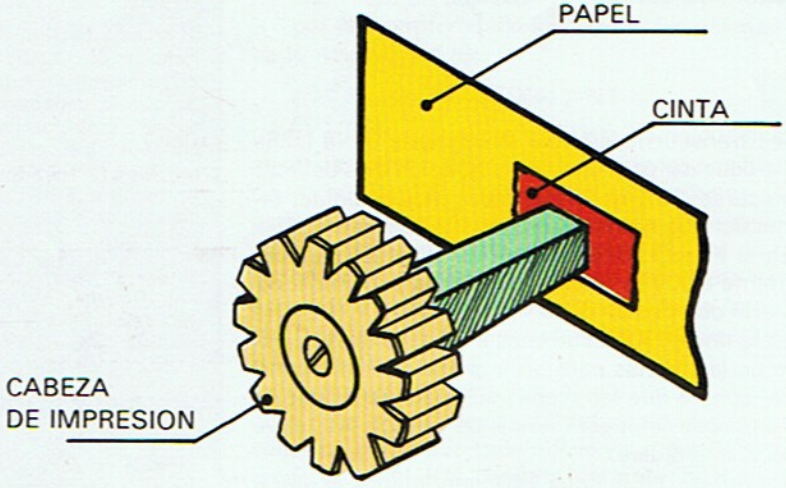
*

Puede obtenerse una variable de cadena de longitud fija, asignando a la variable de principio determinada longitud, y posteriormente dando valor por medio de la apertura y cierre de paréntesis.



IMPRESORAS

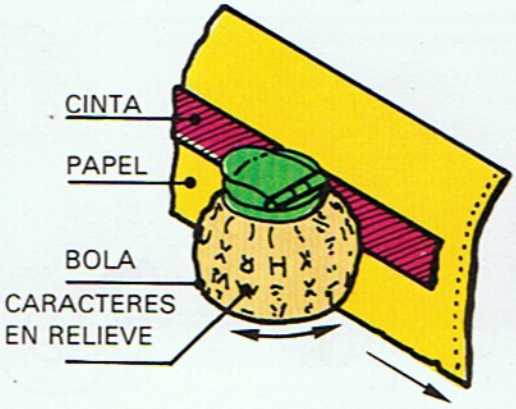
CUANDO nos planteamos la necesidad de utilizar el Spectrum, como una máquina capaz de realizar funciones más avanzadas que la de practicar con nuestros juegos de acción, tarde o temprano, encontramos que se hace fundamental el poder disponer de una copia permanente, en papel, de los listados de los programas que realizamos, así como de sus resultados.
De todos es sabido que la comunicación entre el usuario y el Spectrum, como norma general, se establece entre el teclado y el monitor o pantalla. No es raro, por tanto, que en el momento de realizar la compra de nuestro micro, pensemos que ya hemos resuelto todos nuestros problemas. Nada más lejos de la realidad. En primer lugar, la información que en cada momento muestra la pantalla, va experimentando cambios, y tan solo puede ser visionada de manera temporal. Además, los datos allí expuestos reflejan una situación parcial y es del todo imposible, como no se trate de programas cortos, obtener una visión de conjunto que clarifique el trabajo realizado.
Ciertas aplicaciones como el proceso de textos, balances de producción, contabilidad, etc. no alcanzan su verdadera dimensión hasta el momento de obtener sus resultados impresos en papel. Son las impresoras quienes vienen a solventar todas estas deficiencias.



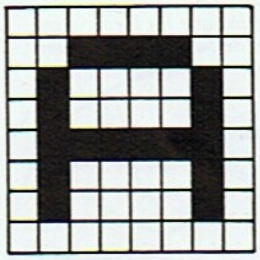
El sistema de impresión matricial es el más difundido entre las impresoras.

TECNICAS DE IMPRESION

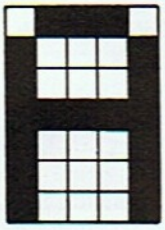
El mercado de los microperiféricos ha experimentado un notable avance desde el auge de los microordenadores y ordenadores domésticos. Nuevos modelos y técnicas de impresión han ido ocupando el lugar de los antiguos terminales impresora, grandes y ruidosos.
De forma general se pueden clasificar las impresoras en dos grandes grupos: de impacto y de no impacto. Las primeras disponen de un elemento mecánico impresor, que incide directamente sobre el papel para imprimir el carácter, como si de una máquina de escribir convencional se tratase. En las de no impacto, por el contrario, no existe elemento mecánico para imprimir.
Dentro de las impresoras de impacto podemos distinguir varios tipos: matriciales o de matriz de puntos, de margarita, de líneas, de banda o cinta, de bola, de cilindro o de tambor. Y las de no impacto se clasifican fundamentalmente en: electrostáticas, térmicas, de inyección y laser.
Otra posible clasificación de las impresoras, sería atendiendo a sus posibilidades gráficas, es decir, a la facilidad para trasladar al papel dibujos o cualquier otro tipo de representaciones. Dentro de este campo, otro factor a considerar es el uso de diferentes colores.



Una vez vistos, a grandes rasgos, los principales mecanismos de impresión, vamos a centrarnos en aquellos que son más frecuentes en las impresoras destinadas a nuestro estimado Spectrum.

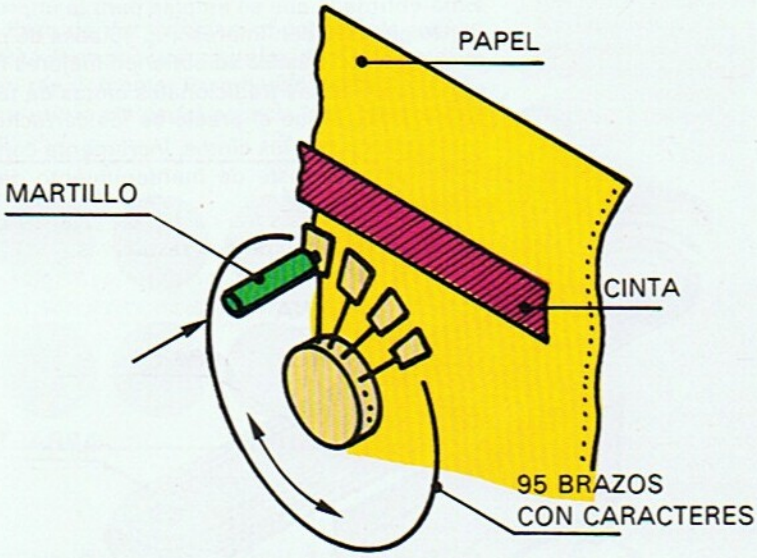


MATRIZ 8 x 8



MATRIZ 5x7

Frecuentemente, el número de puntos que conforman un carácter en la pantalla, no coincide con el asignado en una impresora, debido a lo cual no son iguales los diseños de un mismo carácter en ambos periféricos.

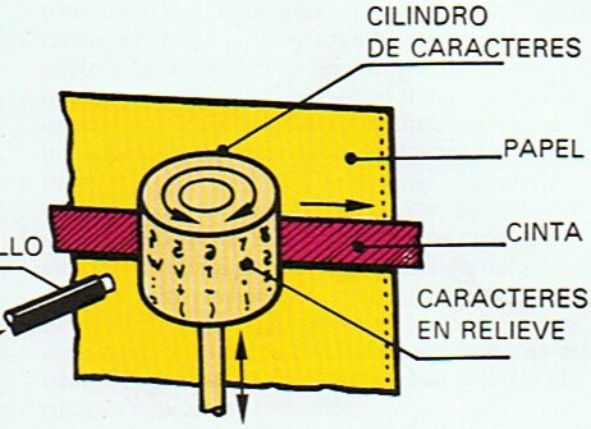


Las impresoras de margarita, aunque muy lentas, obtienen una excelente calidad de letra.

MATRICIALES

Se trata de las impresoras más utilizadas, debido principalmente a su costo reducido, alta fiabilidad y posibilidades gráficas.

Otro tipo de impresoras de impacto, no matriciales, son: de bola (izquierda) y de cilindro (derecha).



La impresión se realiza carácter a carácter. Cada uno de ellos, está formado por una cantidad variable de puntos, según los modelos, que toman posiciones dentro de una matriz. Las más utilizadas son de 7*7 y de 9*9 puntos.

El sistema impresor está formado por una cabeza cilíndrica, con una prolongación hacia el papel. Cuando una serie de solenoides, igual al número de agujas que contiene la cabeza, reciben una descarga eléctrica, activan las agujas, que abandonan su alojamiento haciendo impacto sobre la cinta entintada colocada entre el elemento impresor y el papel.

El carro permanece estático, siendo el elemento impresor el que se desplaza a lo ancho del papel. La impresión de los caracteres puede efectuarse de izquierda a derecha, o en ambos sentidos. Esto se denomina técnicamente modo de impresión unidireccional o bidireccional, que influye, lógicamente, en la velocidad de impresión. La capacidad gráfica de estas impresoras es de tipo medio, ya que los dibujos han de realizarse punto a punto, aunque algunos modelos consiguen resultados bastante buenos. En el caso de impresoras con varias cintas entintadas de distin-

¡!

La norma más comúnmente empleada dentro de los microordenadores, para la transmisión de datos en serie se denomina RS 232-C.

*

A primera vista, el sistema de transmisión en paralelo se puede distinguir porque el cable de conexión con el periférico es plano.

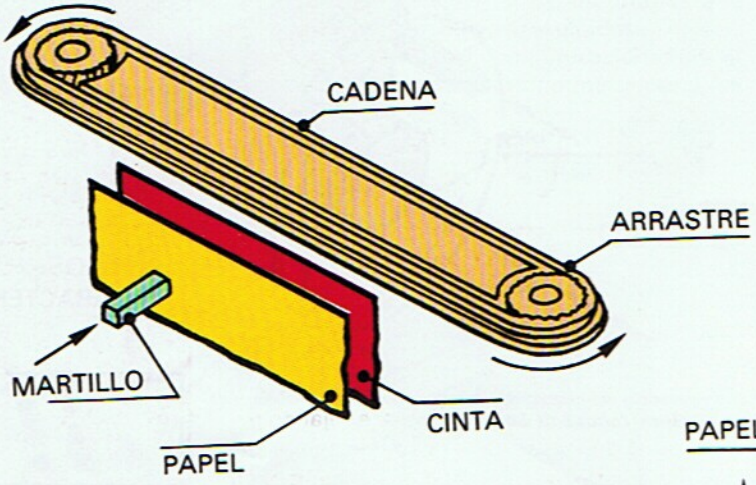
tos colores, los gráficos pueden resultar más que aceptables. En resumen, las impresoras matriciales son adecuadas para procesos en los que no se necesite una velocidad de impresión demasiado elevada, ni sea fundamental una gran calidad de escritura, si bien algunos modelos obtienen ambos efectos.

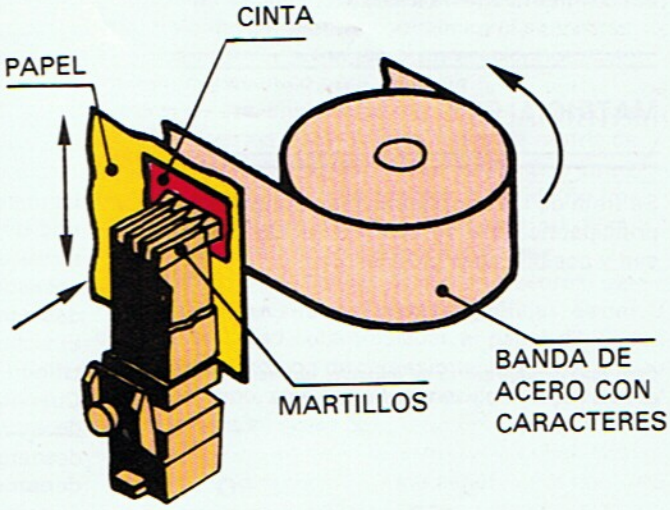
En la calidad de escritura de estas impresoras suele influir de manera considerable el tipo de cinta entintada que se emplee para la impresión. Por lo general, con impresoras dotadas de cartuchos de cinta plástica se obtienen mejores resultados que con las tradicionales cintas de tela, si bien es cierto que el precio de los cartuchos en comparación con las cintas, incrementa considerablemente el coste de mantenimiento del periférico.

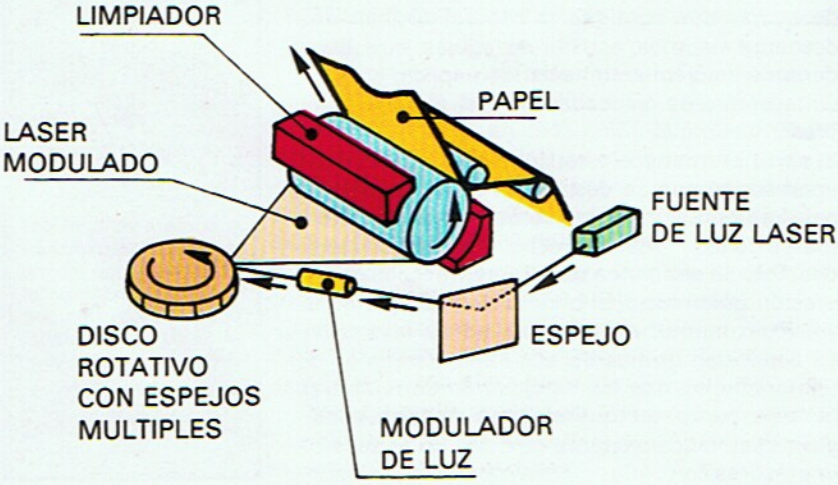
MARGARITA

El elemento principal de estas impresoras es la margarita o tulipa, que da nombre al aparato. Esta es una rueda con 95 brazos (si es estándar), en cuyo extremo se encuentran los caracteres grabados en relieve. Cada brazo de la tulipa contiene un carácter, lo que constituye una limitación.

Algunos sistemas de impresión muy sofisticados y de gran velocidad, están vedados al usuario del Spectrum debido al altísimo precio de los periféricos que los incorporan. Tal es el caso de las impresoras de banda, línea y láser.







No obstante, reemplazar una tulipa por otra con un juego de caracteres distinto es una operación muy sencilla que puede realizar el mismo usuario.
Para efectuar la impresión, el brazo se coloca frente a un martillo que lo impulsa contra la cinta entintada e imprime el carácter sobre el papel. Cada vez que se imprime uno distinto, la tulipa gira hasta posicionar el brazo adecuado frente al martillo, lo que resta velocidad al proceso. En la mayoría de los modelos, la tulipa es capaz de elegir el sentido de giro que le permite situar con mayor rapidez el brazo correspondiente, lo que optimiza un tanto la velocidad de impresión.

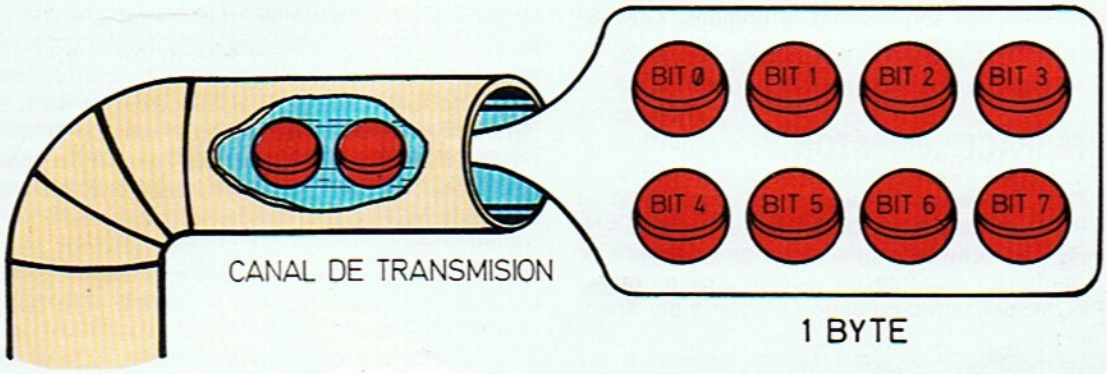
La principal desventaja de estas impresoras frente a las matriciales, es la imposibilidad de realizar gráficos así como su gran lentitud, si bien la calidad de letra es muy superior, al tratarse de un sistema en el que los caracteres se encuentran predefinidos físicamente en la margarita.

INYECCION

El mecanismo es sencillo. El elemento impresor, por lo general, está constituido por un tubo de cristal, en el que circula la tinta, comunicando en la parte posterior con el depósito de ésta. Un orificio en el extremo orientado hacia el papel, permite la salida de la tinta, bien en forma continua, bien como pequeñas gotas que conforman el carácter a base de puntos. En este caso no hay impacto y la densidad de los puntos obtenidos es bastante superior. La calidad de letra conseguida es alta y el nivel de ruido muy inferior, aunque la velocidad de impresión lógicamente disminuye.
En el caso de impresoras multicolor, son tres los depósitos de tinta, que contienen los colores básicos. La mezcla se realiza a la salida del canal de inyección, obteniéndose los colores deseados. El inconveniente de este tipo de impresoras está en la limpieza, ya que al cabo de varias horas de uso, los residuos de la tinta pueden llegar a obturar el cabezal de inyección.

TERMICAS Y ELECTROSTATICAS

Se trata de impresoras silenciosas y económicas. La composición del carácter se realiza usando una técnica esencialmente igual a la de las matriciales.
En el caso de las térmicas, el cabezal de impresión está formado por agujas que se calientan y enfrían rápidamente. Para llevar a cabo la escritura se utiliza un papel especial, sensible al color, sobre el que inciden las agujas reflejando pequeños puntos que forman el carácter.
En las impresoras electrostáticas, las agujas, en vez de calentarse, desprenden descargas eléctricas cuando son activadas, las cuales al incidir sobre un papel especial recubierto de una substancia metálica, reflejan la impresión del carácter. Se trata de impresoras de pequeño tamaño, lo que facilita su transporte, aunque tienden a desaparecer ya que otras técnicas, como la inyección, disponen en la actualidad de modelos reducidos de mayor calidad gráfica.



En el sistema de transmisión de datos en serie, los bytes son enviados bit a bit.

INTERFACES PARA IMPRESORA

El Spectrum tiene dos posibles formas de enviar los datos hacia la impresora:
— Carácter a carácter, denominada transmisión en paralelo, ya que el ordenador envía en cada momento un byte (8 bits, uno al lado del otro).
— Bit a bit (uno detrás de otro), llamada transmisión de datos en serie.
Los cables de interconexión entre ordenador e impresora son diferentes en cada caso. Para la transmisión carácter a carácter se utilizan líneas de cablecitos distribuidos paralelamente (cables planos). Si se trata de enviar datos en serie, el ca-

¡!

Una cualidad muy importante a destacar en las impresoras, es la posibilidad de admitir hojas sueltas (método de alimentación por fricción) y no sólo el tradicional papel continuo o *fan-fold* (método de alimentación por tracción).

*

Recordemos no conectar nunca ningún periférico al bus de expansión del Spectrum, sin haber desconectado previamente ambos aparatos.

*

Los dos sistemas de transmisión de datos para periféricos más importantes son: en paralelo (carácter a carácter) y en serie (bit a bit).

*

La norma más comúnmente empleada dentro de los microordenadores, para la transmisión de datos en paralelo se denomina CENTRONICS.

¡!

Las impresoras matriciales son las más difundidas. Sus características varían en una amplísima gama: desde las de reducido precio y pocas prestaciones, hasta las completísimas, pero de precios prohibitivos para el usuario medio.

*

Al comprar una impresora hay que tener muy en cuenta el coste de su mantenimiento. Fundamentalmente, el de cinta (en caso de que sea matricial) y papel (para térmicas y electrostáticas).

*

A la hora de decidir la compra de una impresora hemos de tener en cuenta las siguientes características:
— Ancho de papel.
— Número de caracteres por línea.
— Sistema de alimentación del papel.
— Velocidad de escritura.
— Tipo de interface.
— Calidad de letra.
— Posibilidades gráficas.

*

Existen en el mercado dos impresoras que no necesitan de interface para funcionar con el Spectrum: la ZX Printer y la Seikosha GP-50. Ambas reducidas en tamaño y... ¡precio!

ble suele ser cilíndrico, con un número inferior de coaxiales en su interior.
Para la transimisión en paralelo, las normas son muy variadas, aunque sin duda la más difundida es la denominada CENTRONICS.
El formato más común en la transmisión de datos en serie es el RS 232-C, que es compatible con la mayoría de los ordenadores y periféricos de entrada/salida. Se le conoce también bajo las denominaciones V 24 o lazo de corriente de 12 mA.
Salvo algunas excepciones, entre las que podemos citar la ZX Printer o la SEIKOSHA GP 50, para acoplar una impresora a nuestro Spectrum es del todo necesario disponer de un interface adecuado. Dos son los más difundidos: KEMPSTON e INDESCOMP.

INTERFACE KEMPSTON

Se trata de un interface del tipo paralelo CENTRONICS. El software de apoyo que necesita va grabado en su interior sobre una memoria EPROM; por tanto, no ocupa memoria del Spectrum.
Lleva incorporadas las funciones **LLIST, LPRINT** y **COPY,** con tres opciones posibles:

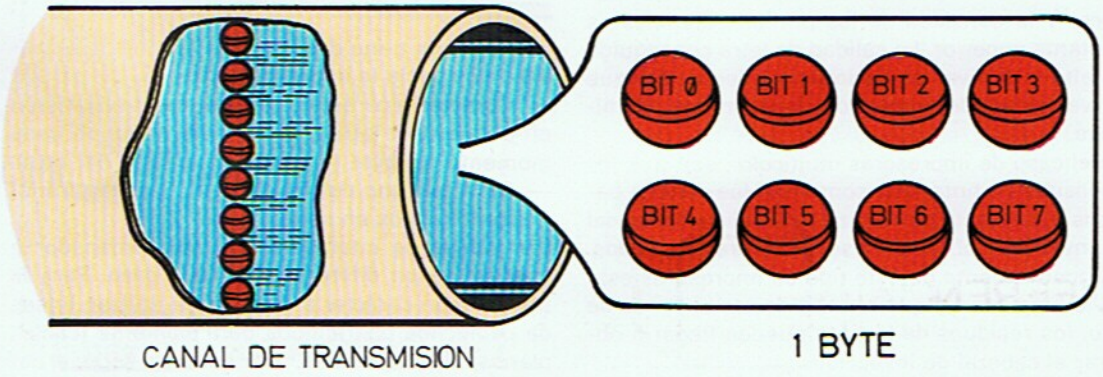
— Modo texto.
— Copia en alta resolución de la pantalla.
— Copia en alta resolución con doble ancho.

Es posible modificar el número de caracteres impresos por línea con POKE 23679,*número de columnas.* Para modificar las numerosas opciones de que dispone el interface, se puede acceder a un menú ejecutando la instrucción **COPY: REM?.** La última línea del menú, es la denominada línea de estado, que indica la situación actual de las opciones modificables del interface.

INTERFACE INDESCOMP

Además de una salida CENTRONICS, incorpora otra del tipo serie RS 232-C bidireccional.Pueden estar conectados a él simultáneamente periféricos distintos, siempre y cuando no ocupen las mismas localizaciones de memoria. Por otra parte, la entrada RS 232 está siempre en activo, pero la salida es intercambiable con la CENTRONICS.
El interface emplea los puertos 251, para comunicación en paralelo, y 127 para transmisión de datos con el RS 232. Como sólo se usa una línea de decodificación, las direcciones cuyos bits 2 y 7 estén a cero, no deben ser utilizadas.
El software se encuentra grabado en EPROM y es volcado en las zonas altas de memoria en el momento de encender el aparato. Este interface es válido solamente para el Spectrum de 48 K y para activar el software hay que ejecutar **RANDOMIZE USR 64973;** proceso de inicialización que habrá de llevarse siempre a cabo después del comando **NEW** y al encender la máquina.
Un conmutador situado en la parte superior, permite seleccionar entre distintas impresoras. En la posición 1 para SEIKOSHA, y en la 2 para Nec, Star, C-Itoh, etc.
Si se utiliza la configuración RS 232, la tasa de baudios, es decir, la velocidad de transmisión de los datos, se puede seleccionar efectuando **POKE**s en las direcciones 64520 y 64521, de un valor entre 50 y 4800.
Para realizar una copia de la pantalla en alta resolución, se debe ejecutar **RANDOMIZE USR 65044.** Esto produce en el papel una imagen de 32*8=256 puntos de ancho. Si se desea una representación de doble anchura, podemos obtenerla con **RANDOMIZE USR 65047,** siempre y cuando la impresora lo admita. Quizás, al no disponer el periférico de las dimensiones suficientes en el carro, se pierdan las últimas columnas de la pantalla.
Finalmente, hemos de añadir que en algunas impresoras, como EPSON, STAR, NEW PRINT, etc., será necesario ejecutar previamente al **COPY: POKE 65524,4;** ya que los gráficos son tratados de forma algo diferente.

Los datos son enviados por bytes en el sistema de transmisión en paralelo.



PROGRESIONES ARITMETICAS

PROGRESIONES ARITMETICAS es el primer programa de aplicaciones matemáticas que introducimos en nuestra Obra. Con ello intentamos no encasillar esta sección de programas en la clásica línea de los juegos, y así salirnos ligeramente de la tónica que veníamos manteniendo hasta el momento.

Este programa está especialmente dedicado a aquellos de vosotros que, dentro de las matemáticas, estéis estudiando el tema de las progresiones; pero antes de introducirnos en los pormenores del mismo, vamos a intentar explicar lo más brevemente posible qué son las progresiones aritméticas.

Una progresión aritmética es una sucesión de números, tales que cada uno de ellos se obtiene del anterior sumándole una constante llamada DIFERENCIA o (impropiamente) razón de la progresión.

Si la diferencia es positiva, la progresión es creciente, porque cada término es mayor que el anterior; como por ejemplo la serie: 1, 3, 5, 7, 9, 11, 13, en que la diferencia es evidentemente 2. Por el contrario, si la diferencia es negativa, la progresión será decreciente, porque cada término es menor que el anterior: 1, –1, –3, –5, –7, –9, –11,... (diferencia –2).

Una vez que hemos estudiado el significado del término PROGRESIONES ARITMETICAS, vamos al punto en que el ordenador puede servirnos de gran ayuda: la resolución de problemas. Los datos o incógnitas que se pueden facilitar o hallar en un problema de progresiones son:

1. A1(Primer término de la progresión).
2. An(Término n-simo o último término de la progresión).
3. N (Número de términos de la progresión).
4. D (Diferencia de la progresión).
5. M (Número de términos a interpolar).
6. Tc (Término central de la progresión).
7. Sn(Suma de los «n» términos de la progresión).

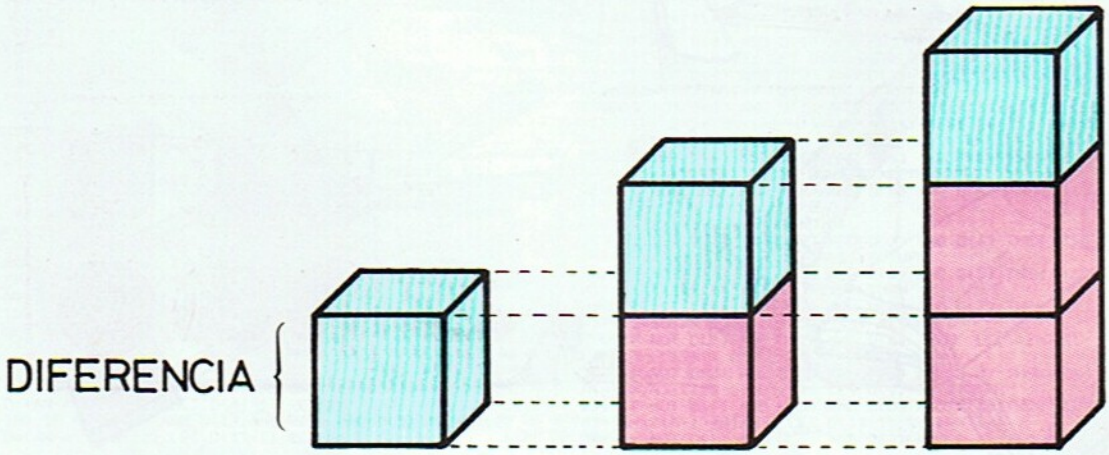
La interpolación de «M» términos (medios diferenciales), consiste en introducir entre dos elementos cualesquiera esos «M» medios diferenciales. Así por ejemplo, supongamos que queremos interpolar 5 medios diferenciales entre 1 y 7; lógicamente los 5 términos que queremos interpolar son: 2, 3, 4, 5 y 6 donde A1=1, An=7 y D=1. El término central (Tc) solamente existe cuando el número de términos de la progresión es impar. En la progresión: 1, 2, 3, 4, 5, 6 y 7 existe término central porque el número de elementos es impar: N=7. Por tanto Tc=4.

Con los datos que hemos visto hasta ahora, cualquiera de nosotros podrá resolver un problema que se le plantee de este tipo. Prácticamente podemos asegurar que todos los datos e incógnitas que pueden aparecer en un ejercicio de este tipo, están contemplados por el programa.

En una progresión aritmética, la diferencia entre cualquier término y el anterior o posterior es constante.

El mayor dato que puede ser introducido no ha de exceder las 10 cifras. En caso contrario el ordenador ignoraría el exceso.

Si los datos introducidos son erróneos (datos falsos para la existencia de una progresión) los resultados son igualmente falsos.

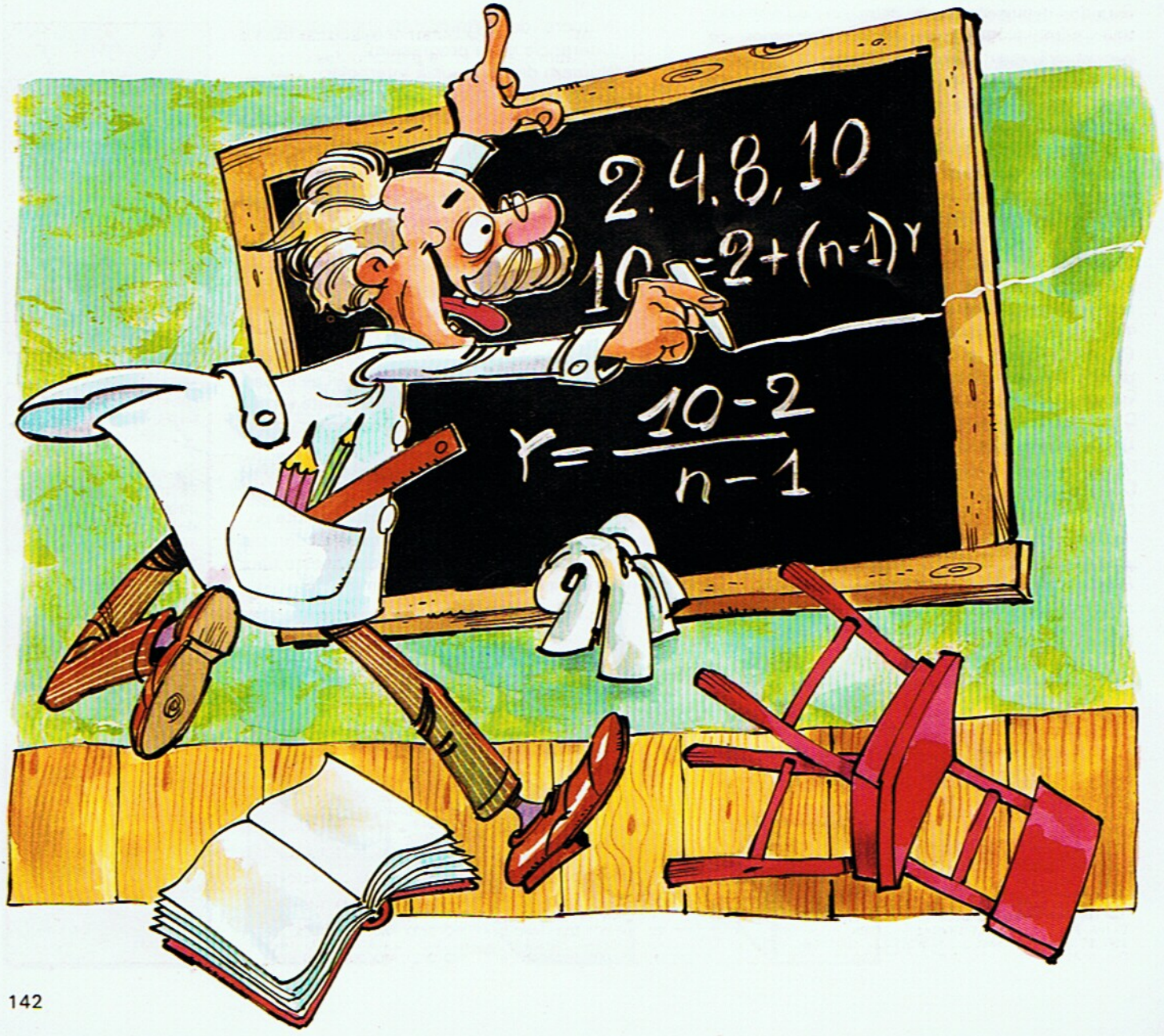


EL PROGRAMA

Nuestro programa utiliza cuarenta y ocho fórmulas para la completa resolución de un problema. Muchos de nosotros nos preguntaremos cómo es posible que existan tantas fórmulas: sí, realmente sólo hacen falta cuatro para que cualquiera de nosotros, con papel, «boli» y un poquito de inspiración matemática podamos calcular todas las incógnitas solicitadas en un problema, basándonos en dichas fórmulas. Sin embargo, el Spectrum, al igual que todos los ordenadores, carece de esa inspiración o intuición que lleva innato el ser humano, y es incapaz de hacer uso de la fórmula o fórmulas ADECUADAS, formando un sistema de ecuaciones si fuese necesario. Precisamente por esto, el ordenador ha de tener muy simplificada toda la información que pueda necesitar.

Cada una de las cuarenta y ocho fórmulas que emplea el programa, están en función de una serie de parámetros. El ordenador utilizará la más adecuada para la resolución de las incógnitas a hallar, dependiendo siempre de los datos que hayamos introducido.

El número mínimo de datos que hemos de facilitar, nunca ha de ser inferior a tres; de lo contrario, el Spectrum nos dirá que le faltan datos. Aún así, en algunas ocasiones, aparecerá en la pan-



talla el mensaje citado, síntoma de que el ordenador se ve incapacitado (al igual que lo estaríamos nosotros con nuestro «boli» y papel) para ejecutar con destreza matemática la misión encomendada.

No obstante, existe un solo caso en el que habiendo introducido dos datos (A1 y An), el Spectrum calcula el término central de la progresión (en caso de que se haya pedido). Dicho resultado existe, pero no es válido al no poderse calcular N (número de términos) y/o D (diferencia) por falta de datos.

El número de término de una progresión aritmética ha de ser lógicamente un número entero y mayor que cero, para que exista dicha progresión. Si por el contrario, N fuese negativo o cero (a causa de los datos erróneos, que con buena o mala fe hayamos introducido) nuestro Spectrum nos lo hará saber.

Otra de las cualidades del programa es que cada una de las incógnitas calculadas durante la ejecución del mismo, puede ser utilizada por el Spectrum como un nuevo dato para hallar, si fuera necesario, otra nueva incógnita.

De todas las variables que se han definido en el listado del programa, las que con más asiduidad consulta el Spectrum son los catorce elementos de lista definidos en los vectores D (7) e I (7). Estos catorce «casilleros» son indicadores, que en todo momento señalan al ordenador qué variables son datos y cuáles son incógnitas a calcular. D (7) es el vector de datos e I (7) el de incógnitas. Cada uno de los elementos de estas listas corresponde a un parámetro en particular:

D (1) & I (1) Indicador de A1
D (2) & I (2) Indicador de An
D (3) & I (3) Indicador de N
D (4) & I (4) Indicador de D
D (5) & I (5) Indicador de M
D (6) & I (6) Indicador de Tc
D (7) & I (7) Indicador de Sn

Siendo K cualquiera de los elementos de las tablas, el Spectrum sabrá que el parámetro correspondiente al número que ostenta K es dato si D (K)=1, o no lo es si D(K)=0. De la misma forma, si es incógnita, I (K) valdrá 1, y de no ser así 0. El ordenador tiene en cuenta en el tratamiento de estas listas que aunque una serie de parámetros no sean datos, no implica que forzosamente sean incógnitas, de ahí que sea necesario definir el vector I, además del D.

TRABAJANDO CON EL PROGRAMA

Una vez introducido correctamente el programa y ejecutado, aparecerán en la pantalla dos cuadros de valores: el de la izquierda hace alusión a los datos, y el de la derecha a las incógnitas.

Por otra parte, en la zona inferior de la pantalla se imprimirá el mensaje "INTRODUCE DATOS", y dos filas más abajo el símbolo (mnemotécnico) del dato a introducir, seguido de su significado real. Como la aparición de cada símbolo es secuencial, solamente tendremos que pulsar **ENTER** si no disponemos de ese dato, o si ese dato existe, teclear el mismo seguido de **ENTER**. El siguiente paso a efectuar es informar al Spectrum sobre cuáles son las incógnitas. Al igual que en la introducción de datos, los símbolos de los parámetros a hallar aparecen en la pantalla secuencialmente. Solamente tendremos que pulsar "S" (sí) para que nuestro Spectrum calcule el parámetro marcado, o "N" (no) para que no lo haga. Ya sólo nos queda esperar unos segundos para ver marcados en el cuadro de la derecha los resultados.

¡!

El programa carece de gráficos definidos, por lo que no aparecerá en el listado ningún carácter subrayado.

*

La grabación del programa debe llevarse a cabo mediante el comando **SAVE**. Si utilizamos el sistema de autoejecución, ésta ha de comenzar a partir de la línea 10. **(SAVE "PROG.ARIT." LINE 10).**

*

Las subrutinas han sido situadas al principio del programa, para conseguir un más rápido acceso a las mismas.

```
 10 REM ************************
 11 REM * J.M.MAYORAL SERRANO *
 12 REM ************************
 20 POKE 23658,8: GO TO 850
 30 REM  TERMINO ENESIMO AN
 40 IF NOT D(2) AND D(1) AND D(3) AND D(4) THEN  LET
AN=A1+(N-1)*D: LET D(2)=1: RETURN
 50 IF NOT D(2) AND D(1) AND D(3) AND D(7) THEN  LET
AN=(2*SN/N)-A1: LET D(2)=1: RETURN
 60 IF NOT D(2) AND D(1) AND D(4) AND D(5) THEN  LET
AN=A1+(M+1)*D: LET D(2)=1: RETURN
 70 IF NOT D(2) AND D(1) AND D(6) THEN  LET AN=(2*TC
)-A1: LET D(2)=1: RETURN
 80 IF NOT D(2) AND D(3) AND D(4) THEN  LET AN=(SN/N
)+(((N-1)*D)/2): LET D(2)=1: RETURN
 90 IF NOT D(2) AND D(6) AND D(3) AND D(4) THEN  LET
AN=TC+(((N-1)*D)/2): LET D(2)=1: RETURN
100 IF NOT D(2) AND D(6) AND D(5) AND D(4) THEN  LET
AN=TC+(((M+1)*D)/2): LET D(2)=1: RETURN
110 IF NOT D(2) AND D(7) AND D(3) AND D(4) AND D(5)
THEN  LET AN=(SN/N)+(((M+1)*D)/2): LET D(2)=1: RETURN

130 REM  RAZON D
140 IF NOT D(4) AND D(1) AND D(2) AND D(3) THEN  LET
D=(AN-A1)/(N-1): LET D(4)=1: RETURN
150 IF NOT D(4) AND D(1) AND D(2) AND D(5) THEN  LET
D=(AN-A1)/(M+1): LET D(4)=1: RETURN
160 IF NOT D(4) AND D(1) AND D(2) AND D(7) THEN  LET
D=(AN-A1)*(A1+AN)/((2*SN)-A1-AN): LET D(4)=1: RETURN
170 IF NOT D(4) AND D(1) AND D(3) AND D(6) THEN  LET
D=2*(TC-A1)/(N-1): LET D(4)=1: RETURN
180 IF NOT D(4) AND D(2) AND D(3) AND D(6) THEN  LET
D=2*(AN-TC)/(N-1): LET D(4)=1: RETURN
190 IF NOT D(4) AND D(1) AND D(3) AND D(5) AND D(7)
THEN  LET D=2*(SN-(A1*N))/(N*(M+1)): LET D(4)=1: RETU
RN
200 IF NOT D(4) AND D(1) AND D(5) AND D(6) THEN  LET
D=2*(TC-A1)/(M+1): LET D(4)=1: RETURN
210 IF NOT D(4) AND D(2) AND D(5) AND D(6) THEN  LET
D=2*(AN-TC)/(M+1): LET D(4)=1: RETURN
230  REM  PRIMER TERMINO A1
240 IF NOT D(1) AND D(2) AND D(3) AND D(4) THEN  LET
A1=AN-((N-1)*D): LET D(1)=1: RETURN
250 IF NOT D(1) AND D(2) AND D(4) AND D(5) THEN  LET
A1=AN-(M+1)*D: LET D(1)=1: RETURN
260 IF NOT D(1) AND D(2) AND D(3) AND D(7) THEN  LET
A1=((2*SN)/N)-AN: LET D(1)=1: RETURN
270 IF NOT D(1) AND D(2) AND D(6) THEN  LET A1=2*TC-
AN: LET D(1)=1: RETURN
280 IF NOT D(1) AND D(7) AND D(3) AND D(4) THEN  LET
A1=(SN/N)-(((N-1)*D)/2): LET D(1)=1: RETURN
290 IF NOT D(1) AND D(6) AND D(3) AND D(4) THEN  LET
A1=TC-(((N-1)*D)/2): LET D(1)=1: RETURN
300 IF NOT D(1) AND D(7) AND D(3) AND D(4) AND D(5)
```

```
 THEN  LET A1=(SN/N)-(((M+1)*D)/2): LET D(1)=1: RETURN
 310 IF NOT D(1) AND D(6) AND D(4) AND D(5) THEN  LET
 A1=TC-((D*(M+1))/2): LET D(1)=1: RETURN
 330 REM  SUMA n TERMINOS SN
 340 IF NOT D(7) AND D(1) AND D(2) AND D(3) THEN  LET
 SN=((A1+AN)/2)*N: LET D(7)=1: RETURN
 350 IF NOT D(7) AND D(6) AND D(3) THEN  LET SN=TC*N:
 LET D(7)=1: RETURN
 360 IF NOT D(7) AND D(1) AND D(2) AND D(4) THEN  LET
 SN=(AN-A1+D)*(A1+AN)/2*D: LET D(7)=1: RETURN
 370 IF NOT D(7) AND D(3) AND D(1) AND D(4) THEN  LET
 SN=N*(2*A1+(N-1)*D)/2: LET D(7)=1: RETURN
 380 IF NOT D(7) AND D(1) AND D(3) AND D(4) AND D(5)
THEN  LET SN=N*(2*A1+(M+1)*D)/2: LET D(7)=1: RETURN
 400 REM INTERPOLACION M MED.DIF
 410 IF NOT D(5) AND D(1) AND D(2) AND D(4) THEN  LET
 M=((AN-A1)/D)-1: LET D(5)=1: RETURN
 420 IF NOT D(5) AND D(3) THEN  LET M=N-2: LET D(5)=1
: RETURN
 430 IF NOT D(5) AND D(1) AND D(3) AND D(4) AND D(7)
THEN  LET M=(2*(SN-(A1*N))/(D*N))-1: LET D(5)=1: RETU
RN
 440 IF NOT D(5) AND D(1) AND D(4) AND D(6) THEN  LET
 M=(((2*TC)-(2*A1))/D)-1: LET D(5)=1: RETURN
 450 IF NOT D(5) AND D(2) AND D(4) AND D(6) THEN  LET
 M=(((2*AN)-(2*TC))/D)-1: LET D(5)=1: RETURN
 470 REM NUM. TERMINOS PROG. n
 480 IF NOT D(3) AND D(7) AND D(1) AND D(2) THEN  LET
 N=(2*SN)/(AN+A1): LET D(3)=1: RETURN
 490 IF NOT D(3) AND D(1) AND D(2) AND D(4) THEN  LET
 N=((AN-A1)/D)+1: LET D(3)=1: RETURN
 500 IF NOT D(3) AND D(5) THEN  LET N=M+2: LET D(3)=1
: RETURN
 510 IF NOT D(3) AND D(1) AND D(4) AND D(6) THEN  LET
 N=(2*(TC-A1)/D)+1: LET D(3)=1: RETURN
 520 IF NOT D(3) AND D(2) AND D(4) AND D(6) THEN  LET
 N=(2*(AN-TC)/D)+1: LET D(5)=1: RETURN
 530 IF NOT D(3) AND D(1) AND D(4) AND D(5) AND D(7)
THEN  LET N=2*SN/((D*(M+1))+(2*A1)): LET D(3)=1: RETU
RN
 540 IF NOT D(3) AND D(6) AND D(7) THEN  LET N=SN/TC:
 LET D(3)=1: RETURN
 560 REM TERMINO CENTRAL Tc
 570 IF NOT D(6) AND D(1) AND D(2) THEN  LET TC=(A1+A
N)/2: LET D(6)=1: RETURN
 580 IF NOT D(6) AND D(7) AND D(3) THEN  LET TC=SN/N:
 LET D(6)=1: RETURN
 590 IF NOT D(6) AND D(1) AND D(3) AND D(4) THEN  LET
 TC=A1+((N-1)*D/2): LET D(6)=1: RETURN
 600 IF NOT D(6) AND D(2) AND D(3) AND D(4) THEN  LET
 TC=AN-((N-1)*D/2): LET D(6)=1: RETURN
 610 IF NOT D(6) AND D(1) AND D(4) AND D(5) THEN  LET
 TC=A1+((M+1)*D/2): LET D(6)=1: RETURN
 620 IF NOT D(6) AND D(2) AND D(4) AND D(5) THEN  LET
 TC=AN-((M+1)*D/2): LET D(6)=1: RETURN
 630 RETURN
 640 REM INICIALIZACION VAR.
 650 DIM D$(10,10): DIM D(7)
 660 DIM S(7): DIM I(7)
 670 LET COL=1: LET SW=0
 680 LET A1=0: LET AN=0
 690 LET N=0: LET D=0
 700 LET M=0: LET TC=0
 710 LET SN=0
 715 BEEP .3,40
 720 RETURN
 730 REM SUPERVISION D$
 740 FOR P=1 TO LEN D$(K)
 750 IF CODE D$(K)(P)>57 OR CODE D$(K)(P)<31 THEN  BE
EP 1,-10: LET SW=1: RETURN
 760 NEXT P
 770 LET D(K)=1: RETURN
 780 REM  BORRADO MENSAJES
 790 FOR C=0 TO 31
 800 PRINT  PAPER 0;AT 19,C;' '
 810 PRINT  PAPER 0;AT 21,31-C;' '
 820 NEXT C
 830 RETURN
 840 REM
 850 REM  PROGRAMA PRINCIPAL
 860 REM
 870 INK 9
 880 INK 9: BORDER 0: PAPER 0: CLS
 890 GO SUB 640
 900 DATA 121,0,0,115,-121,0,0,-115
 910 RESTORE 900
 920 REM   PANTALLA INICIAL
 930 FOR C=4 TO 130 STEP 126
 940 PLOT C,42
 950 FOR T=1 TO 4
 960 READ X,Y
 970 DRAW X,Y
 980 NEXT T
 990 RESTORE 900
1000 NEXT C
1010 PRINT  PAPER 6;AT 1,4;' DATOS '; PAPER 2;AT 1,18
;' INCOGNITAS '
1020 RESTORE 1090
1030 FOR P=1 TO 14 STEP 2
1040 READ A$,B$
1050 PRINT  PAPER 6;AT P+2,1;A$; PAPER 2;AT P+2,17;A$
1060 PRINT AT P+2,3;' =';AT P+2,19;' ='
1070 NEXT P
1080 REM      DATA
1090 DATA 'A1','  (PRIMER TERMINO DE LA PROG)','An','
 (TERMINO ENESIMO PROGRESION)','N','    (NUMERO DE TE
RMINOS PROG.)','D','      (RAZON DE LA PROGRESION)','
M','    (MEDIOS DIF. A INTERPOLAR)','Tc','      (TERM
INO CENTRAL PROG.)','Sn','   (SUMA DE LOS "N" TERMINO
S)'
1100 REM INTROD.  DATOS
1110 RESTORE 1090
1120 PRINT AT 19,4; FLASH 1; PAPER 5; INK 1;'    INTR
ODUCE DATOS
1130 FOR K=1 TO 7
1140 READ A$,B$
1150 PRINT AT 21,0; INK 6; BRIGHT 1; FLASH 1;A$; FLAS
H 0; BRIGHT 0; INK 9; PAPER COL;B$
1160 INPUT  LINE D$(K)
1170 BEEP .2,40
1180 IF D$(K)<>D$(10) THEN  GO SUB 730
1190 IF SW=1 THEN  LET SW=0: GO TO 1160
1200 LET COL=COL+1: NEXT K
1210 REM INTROD. INCOGNITAS
1220 LET COL=1: GO SUB 780
1230 RESTORE 1090
1240 FOR K=1 TO 7
1250 READ A$,B$
1260 IF D(K)=1 THEN  GO TO 1340
1270 PRINT AT 19,0; FLASH 1; PAPER 2;'  INTRODUCE INC
OGNITAS   (S/N)  '
1280 PRINT AT 21,0; INK 2; PAPER 9; BRIGHT 1; FLASH 1
;A$; FLASH 0; BRIGHT 0; INK 9; PAPER COL;B$
1290 IF INKEY$='S' THEN  BEEP .2,30: GO TO 1320
1300 IF INKEY$='N' THEN  BEEP .2,20: GO TO 1330
1310 GO TO 1290
1320 LET I(K)=1: GO TO 1340
1330 LET I(K)=0: GO TO 1340
1340 LET COL=COL+1: NEXT K
1350 GO SUB 780
1360 REM DISPLAY DATOS & INCOG.
1370 FOR K=1 TO 7
1380 IF D(K)=0 THEN  GO TO 1400
1390 PRINT AT 1+2*K,5;D$(K)
1400 NEXT K
1410 FOR K=1 TO 7
1420 IF I(K)=0 THEN  GO TO 1440
1430 PRINT  INK 4;AT 1+2*K,22;'CALCULAR'
1440 NEXT K
1450 PRINT  PAPER 2; FLASH 1;AT 21,0;'      C A L C U
 L A N D O  !
1460 REM CREACION CAMPOS NUM.
1470 IF D(1) THEN  LET A1=VAL D$(1)
1480 IF D(2) THEN  LET AN=VAL D$(2)
1490 IF D(3) THEN  LET N=VAL D$(3)
1500 IF D(4) THEN  LET D=VAL D$(4)
1510 IF D(5) THEN  LET M=VAL D$(5)
1520 IF D(6) THEN  LET TC=VAL D$(6)
1530 IF D(7) THEN  LET SN=VAL D$(7)
1540 REM   CALCULOS
1550 FOR Q=1 TO 7
1560 GO SUB 30
1570 IF D(1) THEN  LET S(1)=A1
1590 IF D(2) THEN  LET S(2)=AN
1610 IF D(3) THEN  LET S(3)=N
1630 IF D(4) THEN  LET S(4)=D
1650 IF D(5) THEN  LET S(5)=M
1670 IF D(6) THEN  LET S(6)=TC
1690 IF D(7) THEN  LET S(7)=SN
1700 NEXT Q
1710 GO SUB 780
1720 REM   TELON
1730 FOR K=3 TO 15
1740 PRINT  PAPER 2; OVER 1;AT K,19;'  '; OVER 0;'
1750 NEXT K
1760 REM CALC. IMPOSIBLES
1770 FOR K=1 TO 7
1780 IF I(K) AND D(K)=0 THEN  PRINT AT 1+2*K,20; FLAS
H 1;'FALTAN DAT.'
1790 NEXT K
1800 GO TO 1890
1810 REM DISPLAY SOLUCIONES
1820 FOR K=1 TO 7
1830 IF I(K) AND D(K) THEN  GO TO 1860
1840 NEXT K
1850 GO TO 1960
1860 PRINT  PAPER 2; BRIGHT 1;AT 1+2*K,21;S(K)
1870 GO TO 1840
1880 REM CASOS IMPOSIBLES
1890 IF D(3) AND N>0 AND N-INT N=0 AND I(6) THEN  GO
TO 1910
1900 GO TO 1940
1910 LET C=N/2: LET R=C-INT C
1920 IF R=0 THEN  PRINT  PAPER 6; FLASH 1;AT 13,20;'
NO EXISTE ': LET I(6)=0: GO TO 1810
1930 GO TO 1810
1940 IF D(3) AND N<=0 THEN  PRINT  PAPER 6; FLASH 1;A
T 7,23;'N <= 0';AT 9,22;'NO OPERO': GO TO 1960
1950 GO TO 1810
1960 REM MAS PROBLEMAS
1970 LET A$='  ALGUN PROBLEMA MAS ??  '
1980 FOR K=1 TO LEN A$
1990 PRINT AT 21,K+1;A$(K)
2000 NEXT K
2010 IF INKEY$='S' THEN  RUN
2020 PRINT AT 21,2; BRIGHT 1;A$: PAUSE 15
2030 IF INKEY$='N' THEN  PRINT USR 0
2040 PRINT AT 21,2; BRIGHT 0;A$: PAUSE 15
2050 GO TO 2010
```

EVALUADOR DE CADENAS

La creación de programas relativamente complejos, requiere del uso de las funciones BASIC. Ya en el último programa de ejemplo propuesto, hemos utilizado la función BASIC **VAL**, aunque sin describir sus características y modo de empleo.

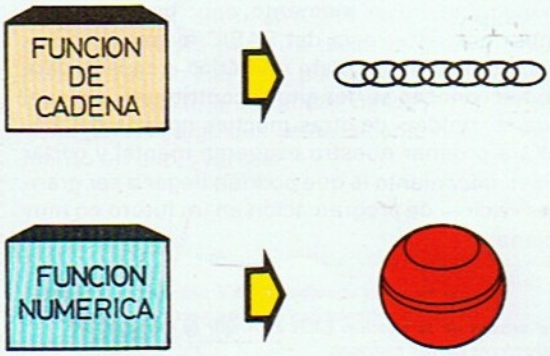
De modo general, podemos decir que las funciones BASIC operan como «traductores», que a partir de un «argumento» que suministramos, producen un valor de salida determinado. De esta forma, la estructura general de cualquier función es:

FUNCION + ARGUMENTO

Como FUNCION, se especifica cualquiera de las múltiples palabras claves de este tipo disponibles en el vocabulario BASIC, y como ARGUMENTO, encerrado entre paréntesis sólo cuando es estrictamente necesario, la expresión a evaluar, que puede ser tan complicada como se quiera; dependiendo de la clase de función elegida, la expresión ARGUMENTO será de tipo numérico o de cadena.

Lógicamente, los argumentos especificados para las diferentes funciones deben ser coherentes con la función escogida, produciéndose, en caso contrario, una detención del programa con el correspondiente mensaje de error. Este será frecuentemente del tipo **C Nonsense in BASIC**, que podemos traducir al Castellano como **C Sinsentido en BASIC** o **C Tontería en BASIC**, si somos más coloquiales y duros con nosotros mismos. En otras muchas ocasiones, los errores de este tipo con funciones, pueden ser ya detectados en la fase de corrección de sintaxis por ese sistema tan eficaz que ya conocemmos: el *syntax checker*.

Las funciones deben operar sobre el tipo de argumento adecuado.



La utilización de una función siempre lleva consigo la presencia de un argumento.

Por los resultados que podemos obtener de ellas, dividimos las funciones en dos grandes grupos, las funciones de cadena y las numéricas. De momento nos ocuparemos de las primeras, que nos permitirán ampliar conocimientos dentro del ámbito del manejo avanzado de cadenas de caracteres, pasando en páginas posteriores al estudio de las funciones de argumento numérico.

Las funciones a que nos referimos en este capítulo son:

STR$, VAL$, VAL, y LEN

De ellas, las dos primeras producen como resultado una cadena de caracteres, mientras que las restantes proporcionan un resultado numérico. Para poder recordar más fácilmente esta circunstancia, basta con advertir que las que producen como resultado una variable de cadena, llevan como sufijo el símbolo dólar **($)**, y las que dan como resultado un valor numérico, no lo llevan, de forma similar a las reglas de construcción de los nombres de variables.

Las funciones pueden admitir como argumento cualquier expresión, siempre que sea del tipo adecuado: de cadena, en este caso. Esto quiere decir que es posible adoptar desde la forma más sencilla, es decir, una constante, hasta nombres de variables o expresiones más complejas por combinación de las anteriores, con inclusión opcional de operadores matemáticos y de otro tipo, o incluyendo otras funciones.

Una parte muy importante de la «inteligencia» de nuestro Spectrum (todo aquello que ya sabe des-

¡!

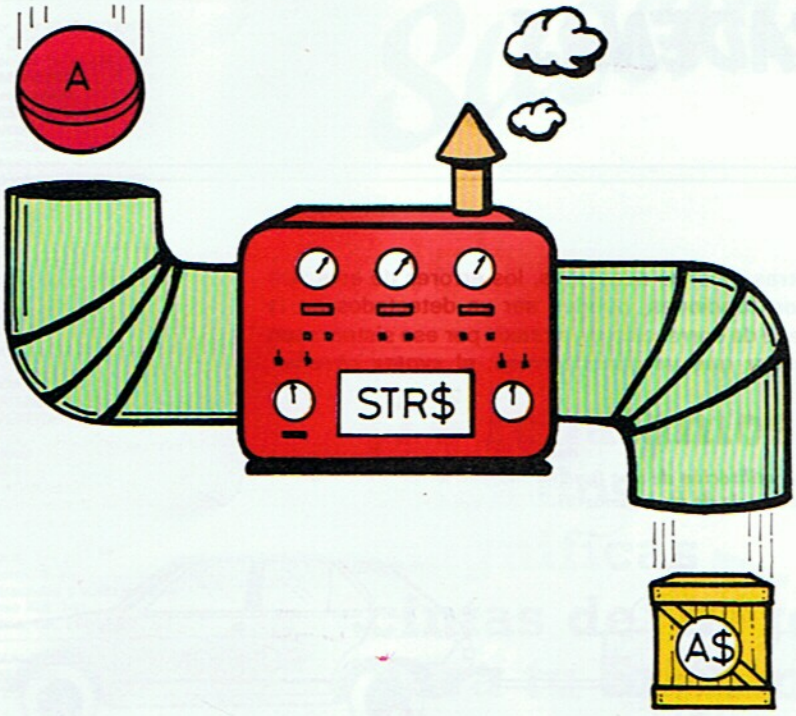
La estructura general de una función es: FUNCION + ARGUMENTO.

*

Los argumentos especificados para las diferentes funciones, deben ser coherentes con la función escogida.

*

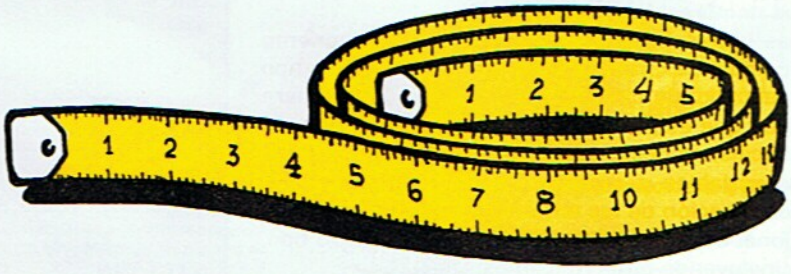
Las funciones destinadas al tratamiento de cadenas en el BASIC son: **STR$, VAL$, VAL** y **LEN**.



La función STR$ *permite convertir las expresiones numéricas en cadenas.*

de que encendemos la máquina, sin que nosotros se los expliquemos), se emplea en facilitar esta evaluación de expresiones, que nos permite, como ya hemos visto, utilizar expresiones auténticamente complejas en prácticamente cualquier punto del programa, y con casi todas las sentencias. Esto, que a los recién llegados a la informática nos parecerá tan natural, no lo es tanto, y se da la paradoja de que versiones del lenguaje BASIC para equipos con un coste más de diez veces superior al del Spectrum, carecen de esta gran facilidad para la utilización de expresiones.

Para cerrar este inciso, brindamos por tanto una cerrada ovación a nuestro Spectrum, y más concretamente al sistema que nos proporciona una comodidad tan considerable en el manejo de expresiones: EL EVALUADOR. Volvemos ahora al tema que nos ocupa.



LEN

LA FUNCION STR$

Esta función permite obtener la forma de cadena de una expresión numérica. En cierta manera, es como convertir una expresión de un tipo al otro, lo cual, aunque a primera vista pueda parecer poco trascendente, es de la mayor importancia, debido a que algunas funciones, como ya hemos comentado, sólo son accesibles para determinados tipos de variables (o expresiones), y gracias a esta circunstancia lo son para todas ellas, previa conversión de la expresión al tipo adecuado para la función.

La impresión de variables o expresiones numéricas BASIC, puede representar un problema cuando lo que se desea es que aparezcan en la pantalla «columnas» de números. Esto es debido a que la sentencia **PRINT** imprime las variables, tanto numéricas como de cadena de izquierda a derecha, mientras que nosotros estamos más acostumbrados a que las expresiones con texto se justifiquen (columnen) por la izquierda y las numéricas por la derecha.

La solución a este problema es sencilla, y se basa en el empleo de la función **STR$**, en combinación con **LEN** y **TAB** (función que estudiaremos más adelante). El motivo de calcular el equivalente en expresión de cadena de una expresión numérica es, precisamente, poder averiguar su longitud, ya que el BASIC no permite conocer la longitud de una variable o expresión numérica de forma directa, sin su conversión previa a expresión de cadena mediante la función **STR$**.

Este hecho nos puede parecer una considerable incomodidad, pero sin embargo se soporta en una sólida base informática; concretamente, en la forma en que la máquina trata y almacena la información numérica. Actualmente no estamos capacitados para comprender el por qué de esto, pero antes de terminar la lectura de nuestra obra conoceremos estas interioridades del BASIC y de la máquina. Por el momento, debe bastarnos con saber que esta regla del BASIC (pasar obligatoriamente cualquier dato numérico a cadena para poder conocer su longitud) contribuirá, al igual que la «rigidez» de otras muchas normas del BASIC, a ordenar nuestro esquema mental y cortar en su nacimiento lo que podrían llegar a ser grandes «vicios» de programación en un futuro no muy lejano.

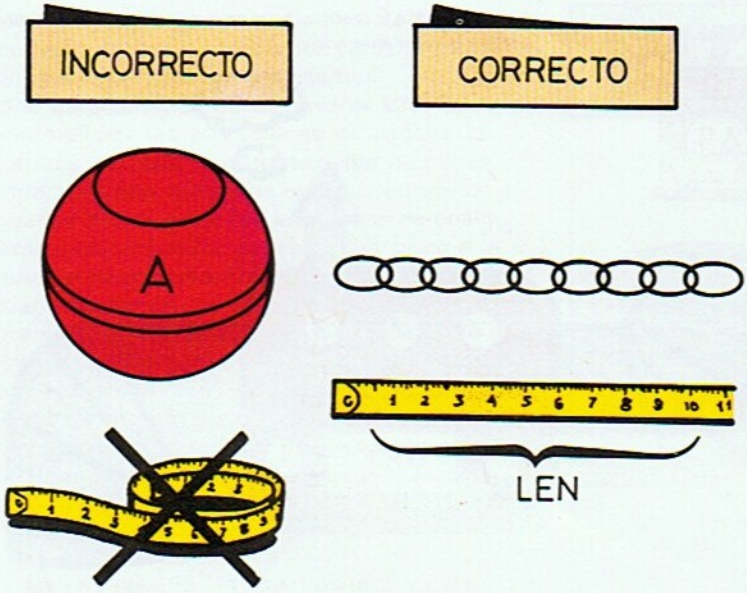
La misión de la función LEN *es medir la longitud en caracteres de las cadenas.*

LA FUNCION LEN

A través de la función **LEN**, podemos calcular el número de caracteres contenidos en una variable o expresión de cadena. En caso de que deseemos conocer el número de caracteres que ocupará la impresión de una variable o expresión de tipo numérico, deberemos calcular primero la **STR$** de la expresión (**STR$** es abreviatura de *STRing,* que como ya sabemos, en inglés significa cadena) para, posteriormente, hallar su **LEN** (abreviatura de *LENgth,* que significa longitud). Vamos a ver más claro el uso de **STR$** y **LEN** con algunos ejemplos:

```
10 REM - STR$ Y LEN
20 LET A=-1234
30 LET B=567890
40 LET A$="ABCDE"
50 LET B$="FGHIJKL"
60 CLS
70 PRINT "Cadena:";STR$ A,"Longitud:";LEN STR$ A
80 PRINT "Cadena:";STR$ B,"Longitud:";LEN STR$ B
90 PRINT "Cadena:";STR$ (A+B), "Longitud:";LEN STR$ (A+B)
100 PRINT "Cadena:";A$,"Longitud:";LEN A$
110 PRINT "Cadena:";B$,"Longitud:";LEN B$
120 PRINT "Cadena:";A$+B$,"Longitud:";LEN (A$+B$)
```

La función **LEN** nos permite calcular la longitud (número de caracteres) de que se compone una cadena, o una expresión de cadena como lo son **A$+B$** o **STR$**. De lo dicho se desprende que el astuto *syntax checker* (comprobador de sintaxis), rechazará olímpicamente cualquiera de nuestros intentos por realizar el **STR$** de una expresión de cadena, cosa por otra parte bastante lógica, puesto que no hay que pensárselo mucho para darse cuenta de la inutilidad de convertir en cadena (**STR$**) algo que ya es cadena; habría poco mérito en ello, ¿verdad?

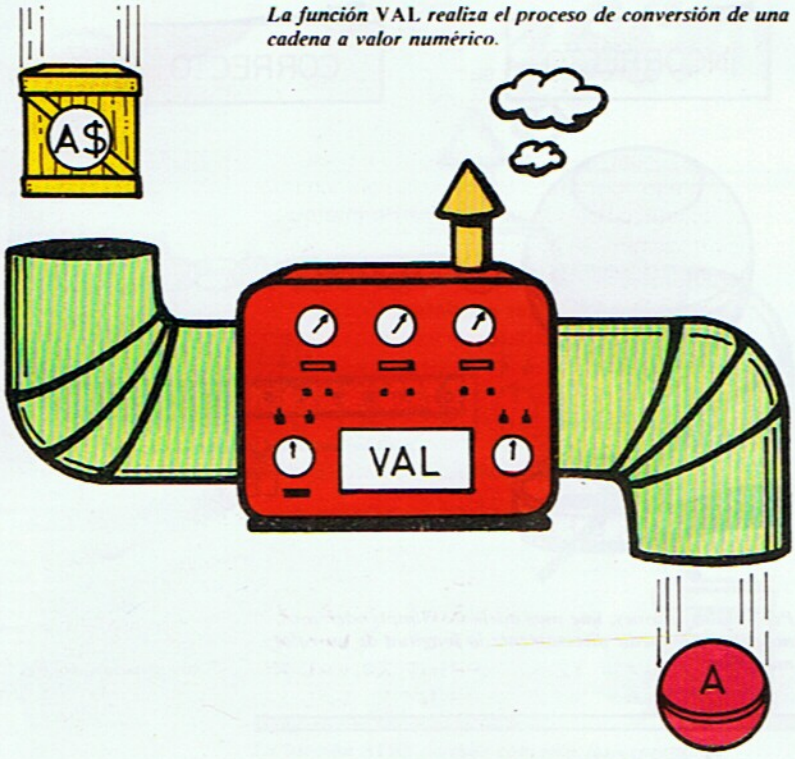


Por sólidas razones, que más adelante comprenderemos, no es posible medir directamente la longitud de un valor numérico.

LA FUNCION VAL

La función **VAL** es, ciertamente, la inversa de la función **STR$**, puesto que calcula el valor numérico de una variable o expresión de cadena. **VAL** es una función de gran potencia, que permite evaluar el contenido de la expresión de cade-

Gracias a la función VAL, podemos emplear nuestro Spectrum como una potente calculadora en la representación de funciones.



La función VAL *realiza el proceso de conversión de una cadena a valor numérico.*

na para calcular su valor numérico. Lo veremos más claro con algunos ejemplos:

```
10 REM - VAL
20 LET A=-1234
30 LET B=567890
40 LET X$="A+B"
50 CLS
60 PRINT X$,VAL X$
```

Hemos comprobado hasta qué punto es capaz la función **VAL** de indagar en el interior de una cadena de caracteres. Todos los elementos que se encuentran dentro de ella, tanto constantes literales como variables y operadores aritméticos, son evaluados para dar como resultado un valor numérico.

A pesar de ello, el caso más común en que se utiliza la expresión **VAL**, es para calcular el valor numérico de una cadena de caracteres, todos ellos numéricos (justo lo inverso a **STR$**).

Sólo con un poquito de imaginación, y habiendo observado la extraordinaria capacidad de análisis de la función **VAL**, debemos intuir que pueden obtenerse los diferentes valores de una función (tabla de valores), que nos permiten dibujar su gráfico. Al fin y al cabo, dada una función matemática cualquiera, por ejemplo *Y=X*X*, hallar la tabla de valores correspondientes para su posterior representación, se limita a ir probando valores de *X* para obtener valores de *Y*; por tanto, ¿qué mejor que la función **VAL** para cumplir ese cometido? Veámosla actuar en el ejemplo propuesto:

```
10 REM - TABLA DE VALORES
20 CLS:PRINT "VALOR DE Y","VALOR DE X"''
30 LET X$="X*X"
40 FOR X=-10 TO 10
50 PRINT X,VAL "X$"
60 NEXT X
```

Esta facilidad en la evaluación de cadenas, que tampoco es frecuente entre otros BASIC, capaces sólo de hallar el **VAL** de constantes, y no de variables o de expresiones, se la debemos agradecer nuevamente al EVALUADOR, puesto que nos proporciona un sistema magnífico para hallar tablas de valores, etc... Sólo tenemos que efectuar por **INPUT** la entrada de la variable de cadena a examinar, y luego **VAL** se ocupará del resto. Así pues, vamos a sustituir la línea 30 del último ejemplo, que asigna a la cadena X$ la función a evaluar, por **30 INPUT "Y= ";X$**, lo que nos va a permitir introducir varias funciones sin necesidad de tener que cambiar cada vez el programa.

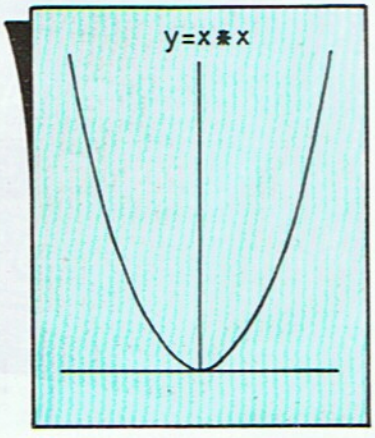
Pero esta facilidad se puede volver en contra nuestra. Si forzamos a calcular el valor numérico de una expresión de cadena en la que está contenido algún carácter no numérico, el Spectrum interpreta que se trata de una variable, y busca en la memoria su valor para reemplazarlo en la expresión. En caso de no encontrarlo, el programa se detiene con el mensaje de error **2 Variable not found** (variable no encontrada).

Esto sucederá en el ejemplo siempre que en la expresión a evaluar introduzcamos variables distintas a *X*, ya que ésta es la que se utiliza en el bucle de las líneas 40 a 60 para buscar los valores de *Y*.

Otros de los problemas a los que nos puede conducir el uso de **VAL**, es a errores del tipo **C Non-**

A partir de la tabla de valores que se calcula en el programa de ejemplo, podemos realizar la representación de la función correspondiente.

x	y
-10	100
-9	81
-8	64
-7	49
-6	36
⋮	⋮
9	81
10	100



sense in BASIC, al intentar evaluar cadenas que no tienen sentido numérico posible; así por ejemplo, un error de este tipo se producirá al ejecutar **PRINT VAL "3*?",** puesto que *3*X* no es una expresión numérica válida. Tengamos en cuenta, que aunque **3** es una constante numérica y el asterisco **(*)** es una operación matemática admitida (multiplicación), el signo de interrogación **(?)** no puede ser considerado ni tan siquiera como parte de una variable.

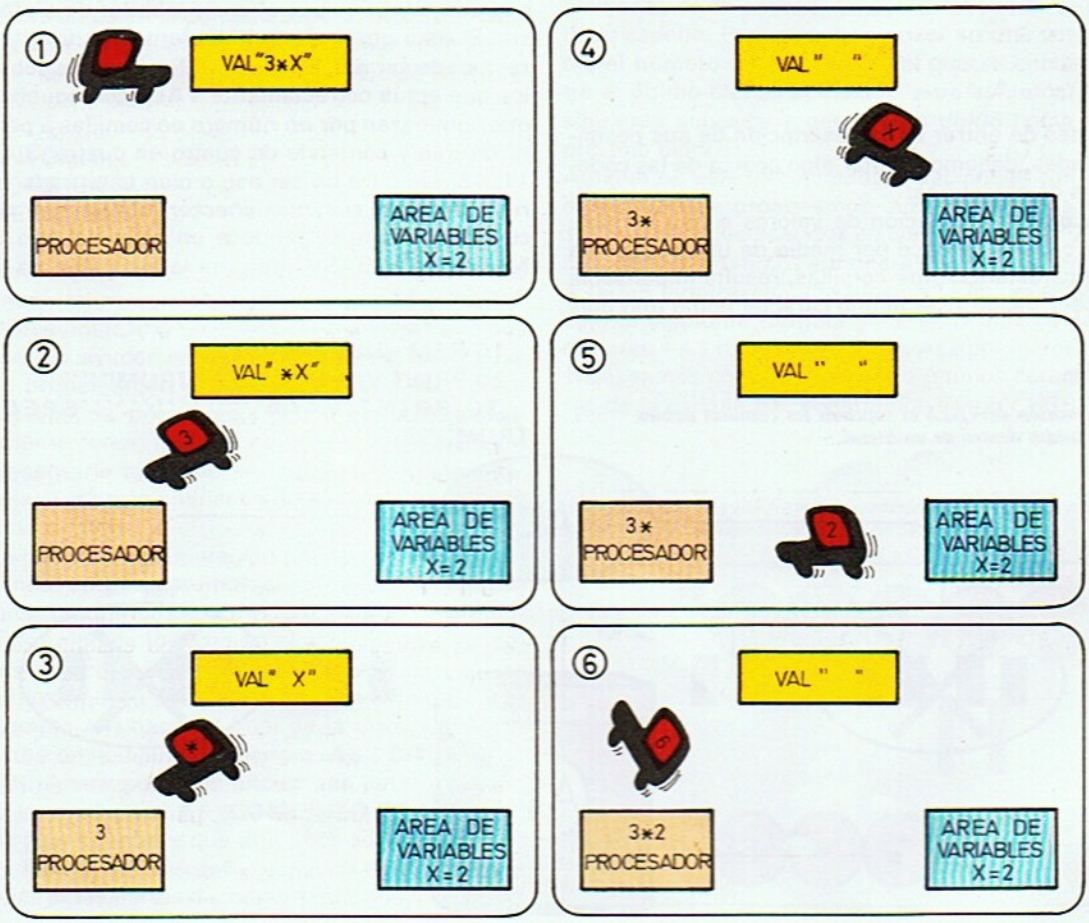
Otros de los problemas que nos pueden surgir con la función que estamos estudiando, son los derivados de la obtención de resultados demasiado grandes o demasiado pequeños, lo que genera errores del tipo **6 Number too big** (número demasiado grande). De entre los problemas de este tipo, probablemente el más frecuente sea el de intentar efectuar divisiones por cero. En nuestro ejemplo, esto puede ocurrir si introducimos la función *Y=X/X,* puesto que en el bucle en el que calcula la tabla de valores, la variable *X* recibe desde –10 hasta 10, con un incremento unitario, pasando por tanto por cero.

El evaluador va algo más lejos que lo expuesto hasta el momento, y es capaz de obtener los valores numéricos de expresiones de cadena que no corresponden exactamente en su formato a una expresión de este tipo. Veamos esto con unos ejemplos; en el formato numérico estándar, los signos aparecen inmediatamente delante del número; del mismo modo, los ceros al comienzo de una cifra nunca se representan.

Una característica de la función **VAL** es la de «normalizar» las expresiones de cadena con el formato estándar de una variable numérica común; para ello, ignora los espacios en blanco que puedan existir entre los elementos de cualquier expresión numérica, así como los ceros a la izquierda de una cifra. Esto nos permite poder efectuar cálculos con variables de cadena, que están preparadas para ser impresas, sin adaptarse concretamente al formato numérico; por ejemplo, con blancos o ceros a la izquierda, sin que ello produzca ningún error. Por otra parte, podemos emplear la función **VAL** para eliminar este tipo de diferencias respecto al formato estándar:

```
10 REM - EJEMPLOS DE VAL
20 CLS
30 LET X$="-   1234":PRINT X$,VAL X$
40 LET X$="-001234":PRINT X$,VAL X$
50 LET X$="0001234":PRINT X$,VAL X$
60 LET X$="- 1234 ":PRINT X$,VAL X$
70 LET X$="    1234":PRINT X$,VAL X$
```

*Diagrama del proceso seguido por el EVALUADOR para el cálculo de la expresión de cadena "3*X".*



BITS

La función **VAL** tiene el cometido inverso al cumplido por **STR$**: calcula el valor numérico de una variable o expresión de cadena.

*

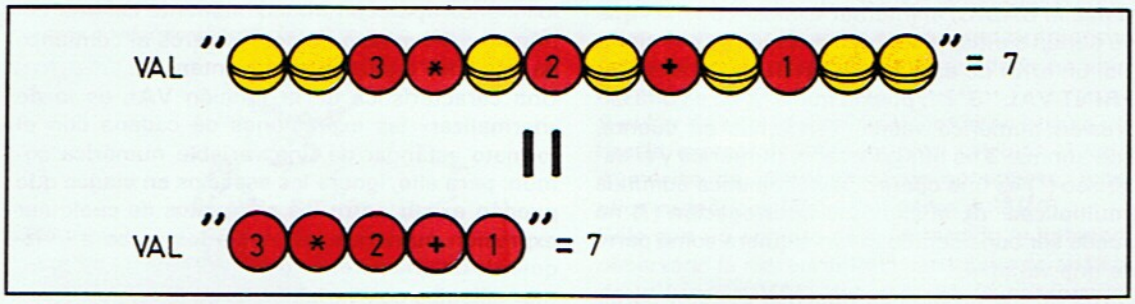
Una característica de la función **VAL**, es la de «normalizar» las expresiones de cadena con el formato numérico estándar.

*

VAL ignora los espacios en blanco que puedan existir entre los elementos de cualquier expresión numérica, así como los ceros a la izquierda de una cifra.

*

La función **VAL$** elimina las dobles comillas de un literal.



La función VAL ignora los espacios en blanco, entre los elementos de una expresión numérica dentro de una cadena.

¡!

Las funciones pueden admitir como argumento cualquier expresión, siempre que sea del tipo adecuado.

*

La función **STR$** permite obtener la forma de cadena de una expresión numérica.

*

Mediante la función **LEN**, podemos calcular el número de caracteres contenidos en una variable de cadena o expresión del mismo tipo.

En este programa se señalan en la parte izquierda de la pantalla, formas muy diferentes que puede adoptar una cifra, y a partir del centro de la misma, la forma normalizada que se consigue mediante el uso de **VAL**.

LA FUNCION VAL$

Antes de entrar en la descripción de sus posibilidades, debemos contar algo acerca de las cadenas de caracteres.

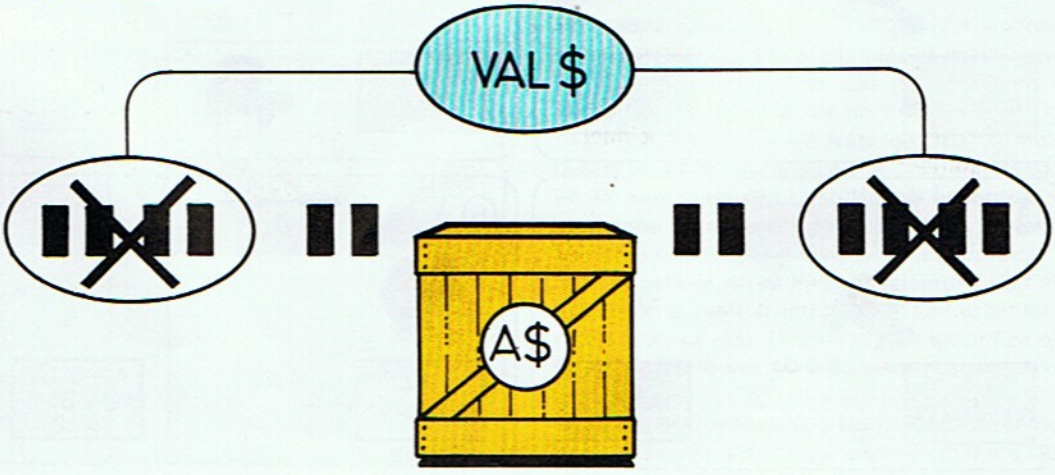
Ya que la asignación de valores a una variable de cadena se hace por medio de una secuencia de caracteres entre comillas, resulta impensable poder incluir este último carácter como uno más dentro de la secuencia que compone la *string*. Existe, en cambio, un método que permite incluir este carácter dentro de la secuencia, que consiste en teclear una doble comilla en el lugar deseado, además de las comillas simples habituales de principio y fin del literal. Este es un ejemplo que muestra cómo incluir una palabra «entrecomillada», dentro de las comillas propias de un literal, asignado a una variable:

LET X$="ESTA ES UNA PALABRA ""ENTRECOMILLADA"" ENTRE COMILLAS"

La función **VAL$**, todo hay que decirlo, es exclusiva del BASIC Sinclair y de muy poco uso. Su misión es eliminar las comillas dobles de una cadena de caracteres, que ha de comenzar y terminar por un número impar de comillas, para que sea admitido por el comprobador de sintaxis. Puesto que las comillas dentro de un literal han de ser dobles, los posibles literales sobre los que actúa correctamente **VAL$** son aquellos que comienzan por un número de comillas a partir de tres y contando de cuatro en cuatro: 3, 7, 11, 15, etc... De no ser así, o bien la entrada es rechazada por el *syntax checker* (número par de comillas) o bien se produce un error del tipo **C Nonsense in BASIC** durante la ejecución de la función:

```
10 REM - USO DE VAL$
20 PRINT VAL$ """"SPECTRUM""""
30 PRINT VAL$ ''''''''''''SPECTRUM''''''''''''
```

La misión de VAL$ es suprimir las comillas dobles incluidas dentro de un literal.



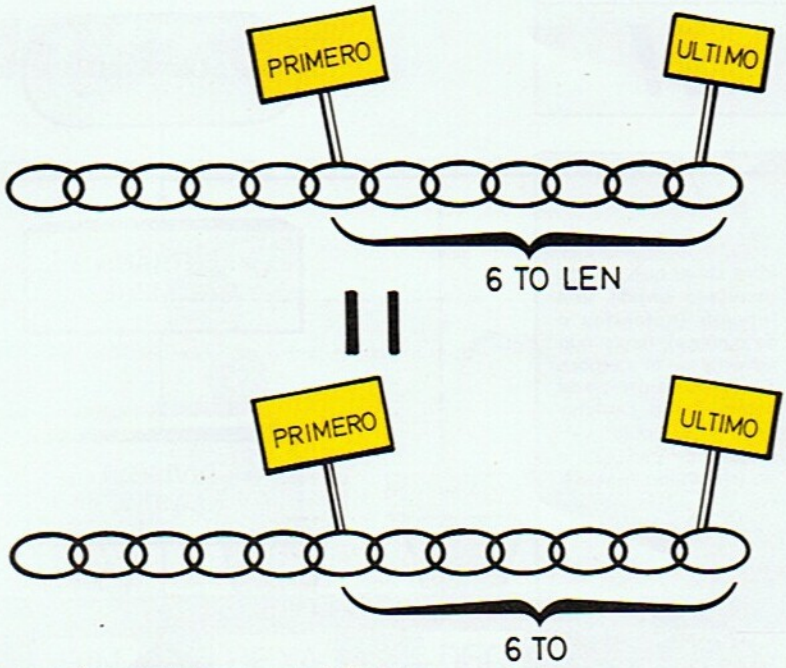
En el ejemplo de la línea 20 obtendremos la palabra sin comilla alguna y en el de la 30 con comillas dobles.
Como podemos observar, cualquier literal debe ir encerrado entre comillas y por tanto, cualquier intento de desproveerle de éstas será rechazado por el ordenador. Así pues, pretender emplear la función **VAL$** sobre un literal sin comillas dobles, abocará a un error del tipo **C Nonsense in BASIC;** tal sería el caso de **PRINT VAL$ "SPECTRUM".**

UN PEQUEÑO REPASO

Con todos los conocimientos que hemos obtenido hasta el momento, ya estamos capacitados para llevár a cabo un ejercicio de relativa complejidad, basado en el tratamiento de cadenas: realizaremos la toma de 10 nombres y los presentaremos en la pantalla separándolos por cada espacio en blanco que encontremos. Por ejemplo, la entrada ANTONIO RODRIGUEZ MENENDEZ, sería escrita:

ANTONIO
RODRIGUEZ
MENENDEZ

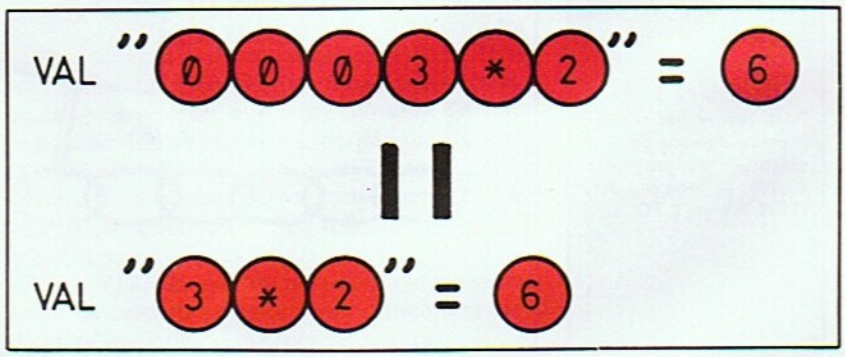
Para separar cada grupo de nombres dejaremos una línea en blanco. Ahora, como es habitual, haremos notar una serie de circunstancias que nos pueden ayudar en la confección del programa. En primer lugar, queda claro que la estructura principal del mismo, es un bucle que realiza la siguiente tarea: toma el nombre y lo separa para presentarlo tal como se requiere y, finalmente, deja el espacio en blanco para diferenciar el próximo nombre.
En cuanto a la separación del nombre en subcadenas, aprovecharemos nuestros conocimientos recién adquiridos sobre la función **LEN** para completar algo de lo dicho en el capítulo anterior sobre el fraccionamiento de *strings;* entonces vimos cómo tomar la parte derecha de una cadena (por ejemplo A$) hasta el final de la misma se efectuaba de la siguiente manera: **A$ (COMIENZO TO).** Ahora podemos añadir que esta expresión es equivalente a **A$ (COMIENZO TO LEN A$).** De esto se infiere que una pista acerca de nuestra manera de buscar los espacios en blanco para separar las partes del nombre entrado, puede ser



Las expresiones que se muestran en el dibujo son equivalentes.

utilizar un bucle cuyo punto inicial sea 1 (primer elemento de la cadena) y cuyo final sea la **LEN** de la cadena. Podemos memorizar en una variable el número de carácter en el que se encuentra el último blanco detectado, para así tomar la siguiente subcadena desde este punto hasta el próximo blanco.
Aunque no esté directamente relacionado con el programa que proponemos, aprovecharemos la ocasión para destacar que en ciertos casos es útil conocer los *n* últimos caracteres de una cadena. A tal fin emplearemos la función **LEN,** operando bajo la siguiente fórmula general: *n* últimos caracteres = A$ (LEN A$–*n*–1 TO). Así, por ejemplo, si deseamos conocer los cuatro últimos caracteres de la cadena A$, deberíamos escribir **PRINT**

La normalización del formato numérico que VAL *lleva a cabo, incluye la supresión de los ceros innecesarios.*



¡!

Para saber qué tipo de resultado brinda una función (numérico o de cadena), basta con advertir si el nombre de la función tiene como último carácter un símbolo dólar (resultado de cadena), o no (resultado numérico).

*

Los posibles literales sobre los que actúa correctamente **VAL$**, son aquellos que comienzan y terminan por un número de comillas a partir de tres y contando de cuatro en cuatro (3, 7, 11, 15...).

*

La expresión VARIABLE$ (COMIENZO TO) es equivalente a VARIABLE$ (COMIENZO TO LEN VARIABLE$).

*

Para tomar los *n* últimos caracteres de una cadena (generalizada a A$), se emplea **A$ (LEN A$-*n*-1 TO)**.

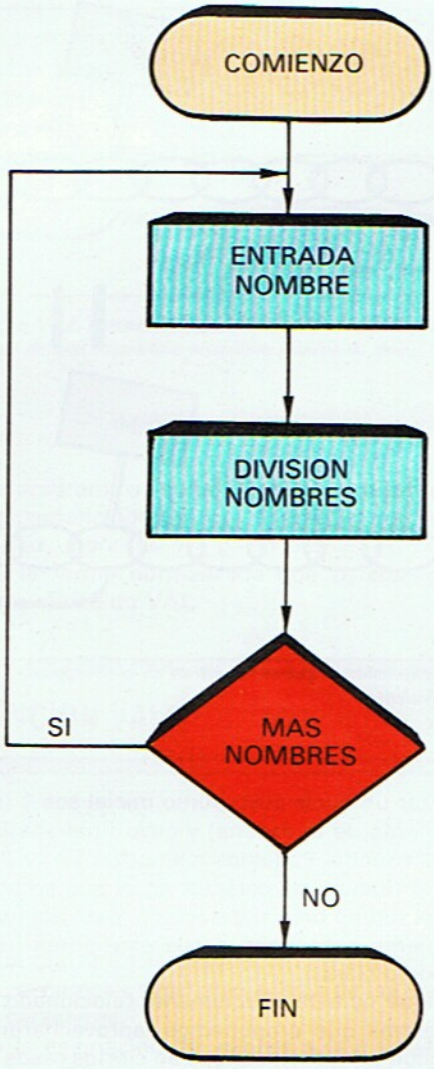


Diagrama de flujo del programa "DIVISION DE NOMBRE".

La función LEN se puede emplear para realizar un slicing inverso, es decir, de derecha a izquierda de una cadena.

A$ (LEN A$-3 TO). Por supuesto, se generará un error en caso de que la cadena A$ contenga menos de cuatro caracteres.

Bien, ha llegado el momento de que intentemos hacer nuestro propio programa:

```
10 REM DIVISION DE NOMBRES
20 CLS
30 FOR I=0 TO 9
40 INPUT "NOMBRE?";N$: LET P=1
50 FOR J=1 TO LEN N$
60 IF N$(J)=" " THEN PRINT N$(P TO J): LET P=J+1
70 NEXT J
80 PRINT
90 NEXT I
```

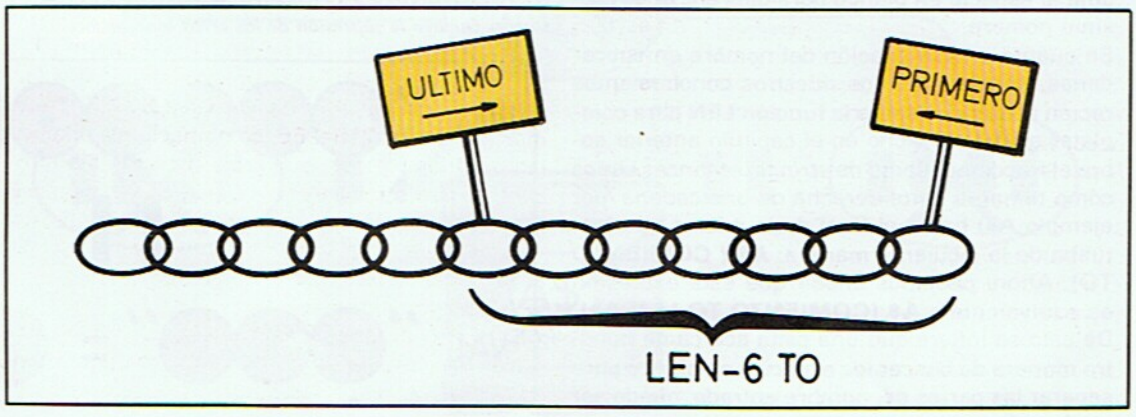
Las líneas 10 y 20 forman la zona de inicializaciones, donde se da título al programa y se borra la pantalla para la presentación más clara de los resultados.

En las líneas 30 y 90 se configura la estructura de bucle basada en la variable I, que consigue que el bloque comprendido entre la línea 40 y la 80 sea repetido 10 veces (de 0 a 9).

En la zona de programa que comprende de la línea 40 a la 80, se realiza el tratamiento de cada nombre. Este comienza por la entrada del mismo a la variable N$ mediante un **INPUT** (línea 40) y finaliza con la impresión de la línea de separación para preparar el tratamiento de un nuevo nombre (línea 80).

Por tanto, la parte más importante de nuestro programa, la cual realiza la división del nombre y la impresión de sus diversas partes, se encuentra en el bucle anidado desde las líneas 50 a la 70. Este bucle de segundo nivel, controlado por la variable J, recorre la cadena entrada (N$) buscando un espacio en blanco (línea 60). Si no lo localiza en el carácter estudiado, continúa analizando el siguiente carácter, y así sucesivamente hasta el final de la cadena.

Si por el contrario localiza un espacio en blanco, escribe la parte de cadena analizada hasta el momento y hace que la variable P indique el último punto tomado como blanco.



SISTEMAS DE NUMERACION

CUALQUIER proceso informático, es decir, de tratamiento de la información, consta básicamente de tres etapas: entrada de los datos, proceso de los mismos y representación de los resultados obtenidos.

Los datos pueden ir desde algo tan simple como dos cantidades a sumar, hasta algo tan complicado como la información obtenida a partir de sensores que midan la humedad del aire, la temperatura, la presión atmosférica, etc., con el objetivo de establecer un pronóstico del tiempo. El proceso consistirá en aquella serie de operaciones que han de ejecutarse, para poder obtener un resultado adecuado a nuestras necesidades.

Al igual que los datos, los resultados podrán ser algo sencillo, como la cantidades suma de las dos anteriores, o algo complicado, como un gráfico de líneas de alta y baja presión, con anticiclones y borrascas, para pronosticar el tiempo de mañana o las próximas semanas.

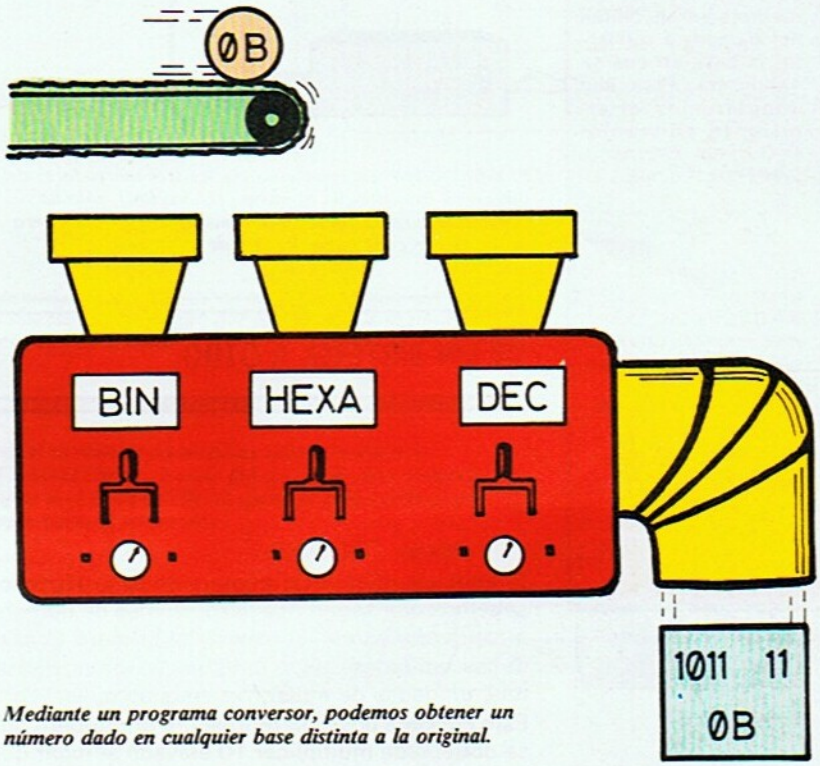
Una mirada a nuestro Spectrum, puede sugerirnos la pregunta de cómo un aparato tan pequeño es capaz de ejecutar infinidad de labores, tan distintas unas de otras. Todo lo anterior, conduce a la idea de ordenador como máquina inteligente, capaz de «pensar» por sí misma. Es más, hasta hace no mucho tiempo, los ordenadores eran denominados cerebros electrónicos.

En realidad, esto no es así. Ni el Spectrum piensa como nosotros, ni es tan inteligente como parece. No es capaz de comprender nuestro lenguaje, y las órdenes que le mandemos ejecutar, antes de almacenarlas en su memoria debe traducirlas a un alfabeto propio, formado únicamente por números.

Ahora bien, nosotros estamos acostumbrados a manejar cantidades numéricas en base 10; es decir, para representar un determinado guarismo, formamos una combinación con las cifras que van del 0 al 9. Ni siquiera eso es capaz de reconocer nuestro micro.

A nadie se le oculta que la electricidad es lo que da vida a nuestro ordenador. Y es aquí donde reside el secreto. Cuando el Spectrum trata de acceder a cualquier información almacenada en la memoria, su particular forma de pensar, sólo sirve para reconocer una cosa: si allí hay corriente o no.

Un convenio adoptado en informática, diferencia entre sí estas dos posibilidades, asignándoles unos valores. Cuando hay corriente, decimos que allí hay un 1. Cuando no la hay, diremos que está a 0. Esto es lo que se denomina lógica binaria, o sistema binario para la representación de la información. Es decir, entre los distintos componentes por los que circula la información dentro de nuestro micro, sólo podemos encontrar dos estados posibles (1 ó 0). Esta es la unidad mínima de información y se denomina BIT *(BInary digiT,* dígito binario).



Mediante un programa conversor, podemos obtener un número dado en cualquier base distinta a la original.

Con un solo bit, no podemos conseguir mucho. Por ello, se asocian en grupos de ocho para formar unidades superiores llamadas BYTES. Tenemos, por tanto, 2↑8=256, combinaciones diferentes de unos y ceros, que representarán distintos caracteres (letras, números, caracteres especiales o de control, etc...). Estas 256 posibilidades, configuran el alfabeto privado del Spectrum. Se trata del denominado código A.S.C.I.I.

Este código es el más utilizado en la comunicación entre periféricos, así como para la representación de caracteres y datos en la memoria central de la mayoría de microordenadores.

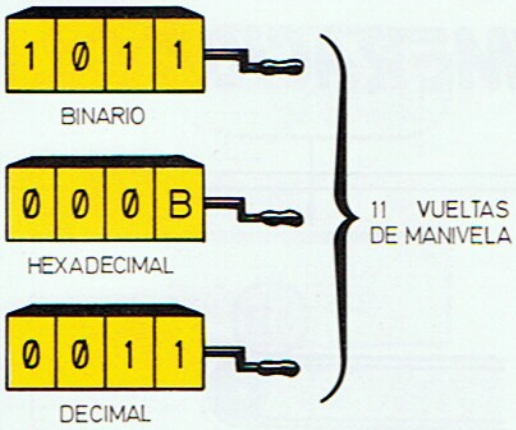
¡!

Para utilizar el programa de conversión, deberemos indicar, inmediatamente detrás del número a convertir, la base en que se encuentra. Para ello adoptaremos el siguiente convenio: D=Decimal, B=Binario, y H=Hexadecimal.

*

Para el cambio de base BINARIA a HEXADECIMAL, podemos emplear una sencilla tabla de conversión:

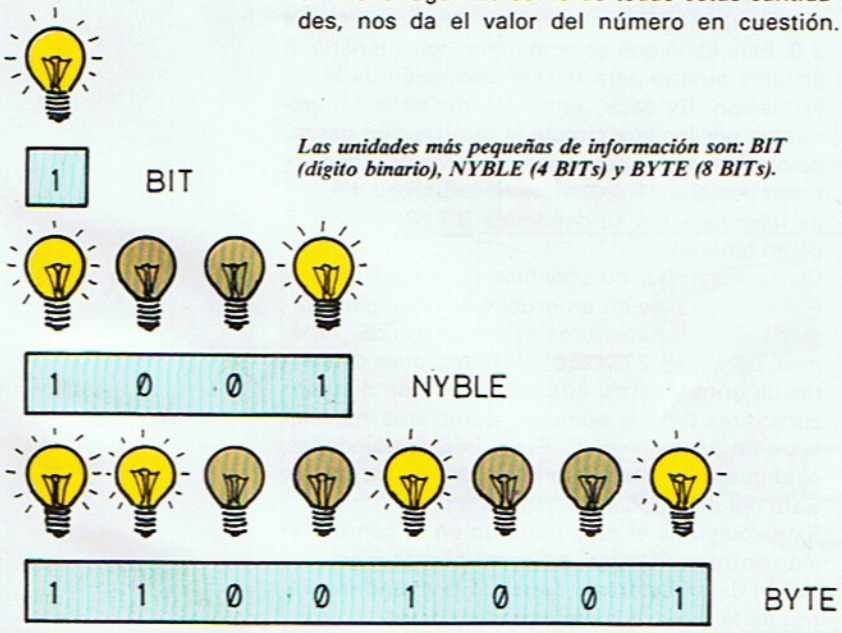
BIN.	HEX.	BIN.	HEX.
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F



El esfuerzo efectuado para mover los tres marcadores del dibujo ha sido siempre el mismo (11 vueltas), sólo ha cambiado la representación del mismo, según el número de dígitos de cada tambor (base de numeración).

DE UN SISTEMA A OTRO

Cuando escribimos un número en base 10, todos sabemos que según nos desplazamos de derecha a izquierda, vamos encontrando primero la cifra de las unidades, luego la de las decenas, centenas, unidades de millar, decenas de millar, etc... Esto es, cada posición representa un número que se obtiene de multiplicar 10 elevado al lugar que ocupa (comenzando a contar desde cero y de derecha a izquierda), por la cifra que se encuentra en dicho lugar. La suma de todas estas cantidades, nos da el valor del número en cuestión.



Las unidades más pequeñas de información son: BIT (dígito binario), NYBLE (4 BITs) y BYTE (8 BITs).

Con un ejemplo concreto lo veremos más claramente. Supongamos que hemos escrito 32956. La cifra 6, ocupa el primer lugar comenzando por la derecha. Tendremos entonces 6*10↑0=6*1=6. El siguiente dígito es un 5. Por tanto, 5*10↑1=5*10=50. El mismo proceso se sigue con los demás: 9*10↑2=9*100=900. 2*10↑3=2*1000=2000 y, finalmente, 3*10↑4=3*10000=30000. Ahora sumamos las cantidades que hemos ido obteniendo: 6+50+900+2000+30000 y el resultado es el número que habíamos tomado en el ejemplo, 32956. Fácil, ¿verdad?

Cualquier otro sistema de numeración, sigue el mismo procedimiento. Lo único que cambiará será la base.

Como vimos antes, el sistema binario o en base dos, tan solo dispone de dos dígitos, el 0 y el 1. Es decir, un número expresado en esta base, estará formado por una combinación de ceros y unos.

Es igual que 1*2↑0

Sea, por ejemplo, el número binario 00101011. Recordemos que la base ahora es 2. Por tanto, comenzando por la derecha tenemos: 2↑0*1=1*1=1, 2↑1*1=2*1=2, 2↑2*0=4*0=0, 2↑3*1=8*1=8, 2↑4*0=16*0=0, 2↑5*1=32*1=32, 2↑6*0=64*0=0 y, para terminar, 2↑7*0=128*0=0. Sumando todos los resultados, 1+2+8+32, hallamos que 43 es el correspondiente valor decimal del número binario 00101011. Como vemos, añadir ceros por la izquierda, no altera el valor del número en cuestión.

La conversión binario-decimal, se efectúa en el Spectrum mediante la función BIN. Si ejecutamos la instrucción **PRINT BIN 00101011** la respuesta será sin duda, 43.

Para realizar el paso contrario, decimal a binario, se utiliza el método de las divisiones sucesivas, entre 2. Es decir, tomamos el número decimal y lo dividimos por dos. Anotamos el resto obtenido. Si el cociente es igual o mayor de dos, volvemos a dividir. Señalamos, de nuevo, el resto. El proceso continúa hasta que resulte el cociente menor que dos. El equivalente binario se construye tomando estos números de abajo a arriba, comenzando por el último cociente, y anotándolos de izquierda a derecha. El cuadro adjunto muestra gráficamente el proceso seguido.

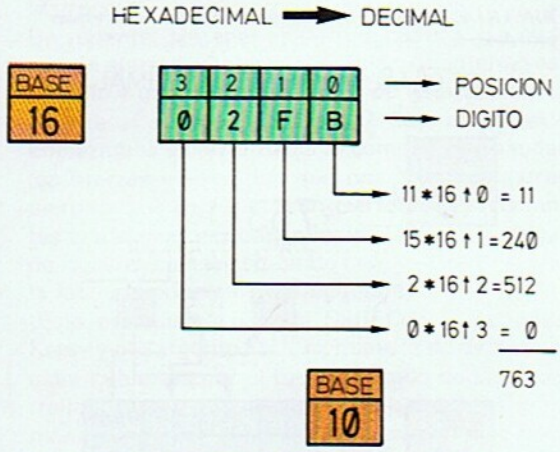
UNA NUEVA BASE

Quizás alguna vez, hayamos asistido atónitos a listados de programas escritos en código máqui-

na, que utilizan una extraña notación, mezcla de números y letras. Y bien, ¿no habíamos quedado en que nuestro ordenador sólo era capaz de identificar dígitos binarios (bits)? Entonces, si el código máquina es lo más aproximado al lenguaje de nuestro Spectrum ¿por qué no está formado por unos y ceros?

La respuesta es sencilla. Imaginemos un listado de miles de instrucciones, compuesto únicamente por ceros y unos. ¡A ver quién es el valiente que interpreta eso!

Para facilitar la labor de introducción del código máquina, acortando la longitud de sus cifras, se emplea otro sistema de numeración, en el que probablemente hemos visto los listados antes citados: el hexadecimal, que expresa los números en base 16. Mediante este sistema de numeración podemos expresar números más grandes con menor cantidad de dígitos. Además, como

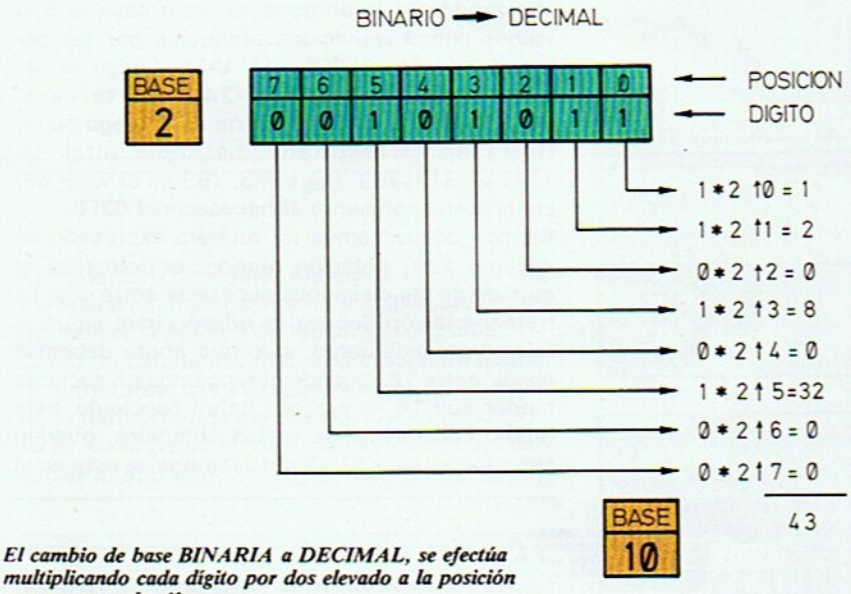


El cambio de base HEXADECIMAL a DECIMAL, se efectúa multiplicando cada dígito por dieciséis elevado a la posición que ocupa en la cifra.

tendremos oportunidad de comprobar más adelante, es muy fácil la conversión del sistema hexadecimal al binario.

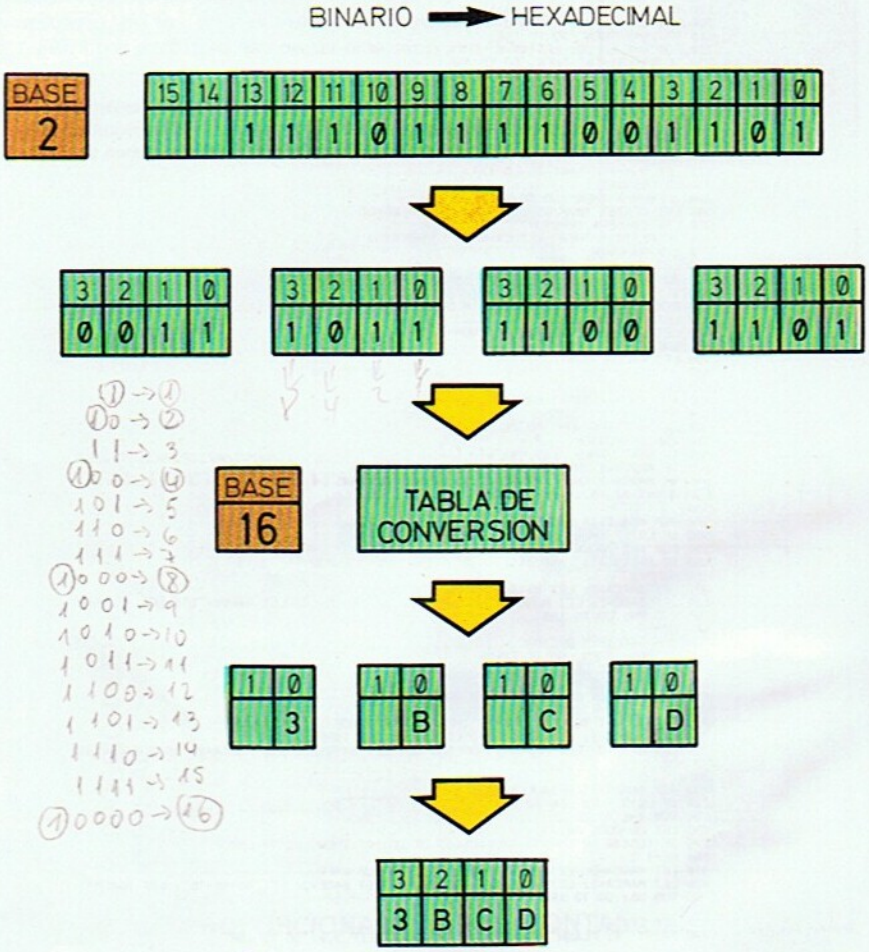
El sistema binario dispone de dos dígitos, 0 y 1. El decimal de 10, del 0 al 9. Por tanto, en hexadecimal, tendremos 16 dígitos, que toman valores de 0 a 15. Y aquí es donde surge el problema. Estamos hablando de dígitos, es decir, una sola cifra para cada posición dentro del número. De 0 a 9, no hay ninguna ambigüedad, pero para representar más de 10 necesitamos dos cifras. La cuestión se solventa asginando a los valores que van del 10 al 15 las primeras letras del alfabeto. Según esto, en vez de 10, escribimos A. El 11, es una B, el 12 una C, el 13 una D, el 14 una E y el 15 una F.

La forma de tratar los números en base 16, es idéntica a como lo hacíamos con los sistemas binario y decimal. Sea, por ejemplo, el número 02FB, en base 16. Tomamos la primera cifra co-



El cambio de base BINARIA a DECIMAL, se efectúa multiplicando cada dígito por dos elevado a la posición que ocupa en la cifra.

Para el cambio de números en base BINARIA a HEXADECIMAL, resulta útil dividir la cifra completa en grupos de 4 BITs (NYBLE), para así poder utilizar una simple tabla de conversión.

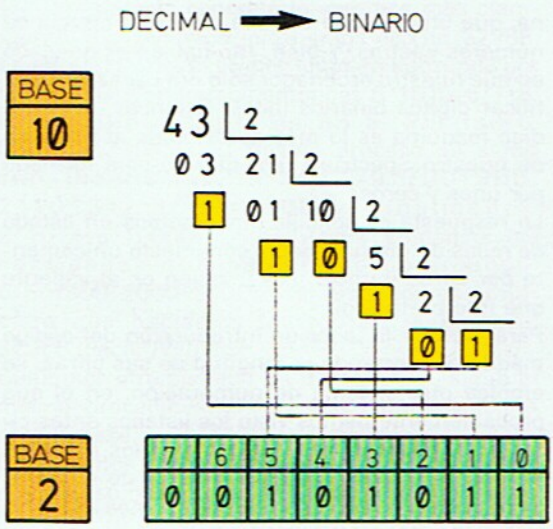


menzando por la derecha, en este caso la B, y vamos construyendo los productos por las potencias de 16: 16↑0*B=1*11=11. Luego la segunda: 16↑1*F=16*15=240. La tercera: 16↑2*2=256*2=512. La cuarta es 0, luego no influirá en el resultado. La suma total es: 11+240+512=763. Por tanto, 763 es el valor decimal correspondiente al hexadecimal 02FB.

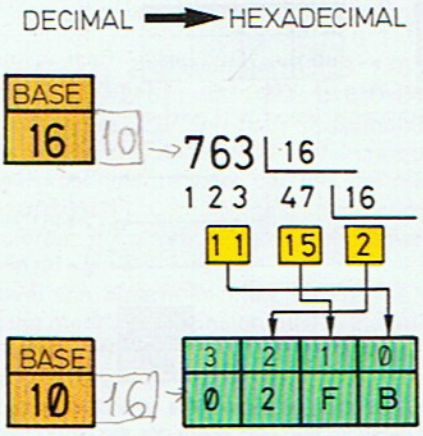
Cuando convertíamos un número expresado en decimal a su notación binaria, utilizábamos el método de las divisiones sucesivas entre dos. La transformación decimal a hexadecimal sigue el mismo procedimiento, sólo que ahora debemos dividir entre 16. Cuando obtengamos un cociente menor que 16, el proceso habrá concluido. Este último cociente o los restos obtenidos, pueden ser superiores a 10. Como sabemos, si este es el caso, sustituiremos dichos valores por las letras correspondientes. En el cuadro se ilustra el proceso seguido.

El paso de binario a hexadecimal y viceversa, se efectúa en dos etapas. En la primera convertimos nuestro número a su equivalente decimal, y desde aquí a la base que deseemos, siguiendo los métodos descritos anteriormente. Es conveniente, cuando se trate de números binarios con gran cantidad de cifras, separarlas en grupos de cuatro, comenzando por la derecha. Si el último grupo tiene menos de cuatro dígitos, añadimos el número de ceros necesarios, a la izquierda, hasta completarlo. El ejemplo del cuadro aclarará las ideas.

El proceso de conversión entre los distintos sistemas, realizado a mano, puede resultar largo y tedioso. Nuestro Spectrum, será una gran ayuda. El siguiente programa realiza todo el trabajo.



Para la conversión de un número en base DECIMAL a BINARIA, se emplea el sistema de divisiones sucesivas por dos.



Para la conversión de un número en base DECIMAL a HEXADECIMAL, se emplea el sistema de divisiones sucesivas por dieciséis.

```
 10 REM HEXA.BIN.DEC $ Carlos de la Osa
 20 BORDER 4: BRIGHT 1: PAPER 4: CLS
 30 PLOT 35,0: FOR n=0 TO 23: IF n=12 THEN  PLOT 40,5
 40 READ x,y: DRAW x,y: NEXT n
 50 DATA 185,0,0,35,35,0,0,185,-35,0,0,35,-185,0,0,-35,-35,0,0,-105,35,0,0,-35
 60 DATA 175,0,0,35,35,0,0,95,-35,0,0,35,-175,0,0,-35,-35,0,0,-95,35,0,0,-35
 70 PLOT 17,17: FOR n=0 TO 7: IF N=4 THEN  PLOT 19,19
 80 READ x,y: DRAW x,y: NEXT n
 90 DATA 221,0,0,141,-221,0,0,-141
100 DATA 217,0,0,137,-217,0,0,-137
110 PLOT 25,45: DRAW 205,0: DRAW 0,85: DRAW -205,0: DRAW 0,-85
120 PLOT 26,46: DRAW 203,0: DRAW 0,83: DRAW -203,0: DRAW 0,-83
130 PRINT AT 3,11; BRIGHT 1;"CONVERSOR"
140 PRINT AT 7,4;"DECIMAL:": PRINT AT 10,4;"BINARIO:": PRINT AT 13,4;"HEXADECIM
AL:"
150 LET R$="              "
160 POKE 23658,8: POKE 23609,20: POKE 23562,1
170 LET DB=280: LET DH=280: LET BD=410: LET HD=410: LET IMPR=510: LET ANB=570:
LET AND=650: LET ANH=750: LET ERROR=840
180 INPUT ;"NUMERO ? "; LINE A$
190 PRINT AT 7,12;R$;AT 10,12;R$+"     ";AT 13,16;R$
200 LET LG=LEN A$
210 IF LG=0 THEN  GO TO 180
220 FOR N=1 TO LG: LET A$(N)=CHR$ ((CODE A$(N)-(32 AND CODE A$(N)>72))): NEXT N
230 IF A$(LG)="B" THEN  GO TO ANB
240 IF A$(LG)="D" THEN  GO TO AND
250 IF A$(LG)="H" THEN  GO TO ANH
260 GO TO ERROR
270 :
280 REM DECIMAL A BINARIO Y HEXADECIMAL
290 LET LG=LEN A$
300 LET N$=""
310 LET NUM=VAL (A$( TO LG-1))
320 LET BS=BASE
330 IF NUM=0 THEN  GO TO IMPR
340 LET CC=INT (NUM/BS): LET RESTO=NUM-BS*CC
350 LET C$=STR$ (RESTO)
360 IF RESTO>9 THEN  LET C$=CHR$ (55+RESTO)
370 LET N$=C$+N$
380 LET NUM=CC
390 GO TO 330
400 :
410 REM BINARIO Y HEXADECIMAL A DECIMAL
420 LET BS=BASE
430 LET D$=A$( TO LG-1): LET NUM=0
440 FOR I=0 TO LG-2
450 LET A=CODE (D$(LG-1-I))
460 IF A>=48 AND A<=57 THEN  LET A=A-48: GO TO 480
470 LET A=A-55
480 LET NUM=A*(BS^I)+NUM
490 NEXT I
500 LET N$=STR$ NUM: GO TO IMPR
510 REM IMPRESION
520 IF N$="" THEN  LET N$="0"
530 IF MARCA=1 THEN  PRINT AT 7,12;N$: RETURN
540 IF MARCA=2 THEN  PRINT AT 10,12;N$: RETURN
550 PRINT AT 13,16;N$: RETURN
560:
570 REM ANALISIS BINARIO
580 IF LG>17 THEN  GO TO ERROR
590 FOR I=1 TO LG-1
600 IF A$(I)<>"1" AND A$(I)<>"0" THEN  GO TO ERROR
610 NEXT I
620 LET MARCA=2: LET N$=A$( TO LG-1): GO SUB IMPR
630 LET BASE=2: LET MARCA=1: GO SUB BD: LET BASE=16: LET A$=N$+"D": LET MARCA=3
: GO SUB DH: GO TO 180
640 REM
650 REM ANALISIS DECIMAL
660 IF LG>6 THEN  GO TO ERROR
670 FOR I=1 TO LG-1
680 LET CG=CODE A$(I)
690 IF CG<48 OR CG>57 THEN  GO TO ERROR
700 NEXT I
710 IF VAL A$( TO LG-1)>65535 THEN  GO TO ERROR
720 LET MARCA=1: LET N$=A$( TO LG-1): GO SUB IMPR
730 LET BASE=2: LET MARCA=2: GO SUB DB: LET BASE=16: LET MARCA=3: GO SUB DH: GO
TO 180
740:
750 REM ANALISIS HEXADECIMAL
760 IF LG>5 THEN  GO TO ERROR
770 FOR I=1 TO LG-1
780 LET CG=CODE A$(I)
790 IF (CG<48 OR CG>57) AND (CG<65 OR CG>70) THEN  GO TO ERROR
800 NEXT I
810 LET MARCA=3: LET N$=A$( TO LG-1): GO SUB IMPR
820 LET MARCA=1: LET BASE=16: GO SUB HD: LET BASE=2: LET A$=N$+"D": LET MARCA=2
: GO SUB DB: GO TO 180
830:
840 REM ERROR
850 PRINT  0; FLASH 1;"ENTRADA ERRONEA": BEEP 3,0: GO TO 180
```

LA BOMBA

POR una de esas «casualidades» de la vida, nos encontramos cumpliendo nuestro Servicio Militar. Estamos encuadrados en la SERECO (Sección de Reconocimiento), perteneciente a la compañía de armas de apoyo del II Batallón del Regimiento de Infantería más aguerrido de toda la Nación.

Aparentemente es un día como otro cualquiera: luce el sol, nos refresca una suave brisa, los hombres del I Batallón piden a voz en grito «más instrucción...»: una jornada muy normal.

De repente, sin apenas darnos cuenta, las sirenas de alarma comienzan a ulular; un intenso escalofrío y una breve sensación de ignorancia nos recorre el cuerpo. ¿Qué pasa? Casi al instante empezamos a correr hacia la compañía con todas las fuerzas y velocidad que nos daban nuestras piernas. Los rumores eran ciertos. ¡¡Los visitantes extraterrestres han invadido la Tierra y rodeado nuestro cuartel, situando cargas explosivas en la fachada principal del Regimiento!!

¡Esto era misión para la SERECO! El Teniente Korsy y el Sargento Multus, mandos de dicha sección, cubiertos por el fuego cruzado de las ametralladoras y subfusiles que con habilidad supina manejaban sus hombres, desactivaron con éxito cada uno de los artefactos que los lagartos extraterrestres habían colocado.

Pero... ¿Cómo lo lograron?, ¿qué método utilizaron para desactivar las claves? Eso es precisamente lo que debemos intentar averiguar con este programa.

COMO JUGAR

El Spectrum «piensa» una palabra y nosotros tenemos que adivinarla, introduciendo en cada secuencia del juego una letra. Debemos tener cuidado, porque por cada intento errado, el brazo extensible del detonador se acercará cada vez más a la mecha de la bomba. En el momento que se falle por octava vez, el artefacto quedará irreversiblemente activado y listo para explosionar.

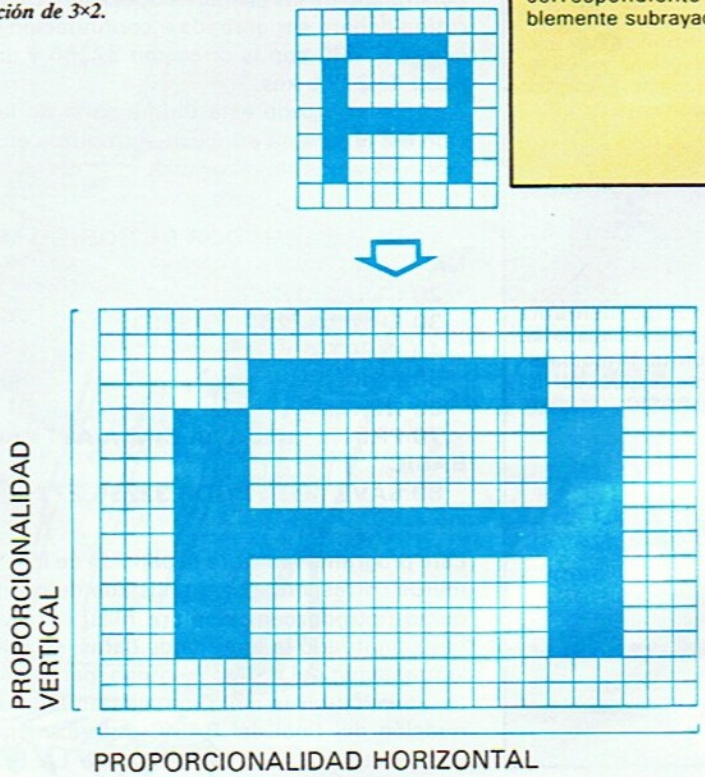
Si no hemos conseguido encontrar la clave, nuestro Spectrum hará gala de su caballerosidad mostrándonos la palabra de nuestras desdichas.

Como un detalle más del programa, para facilitar nuestra tarea, cada vez que realizamos un intento, la letra utilizada queda impresa en la parte inferior de la pantalla, recordándonos que no debe volver a utilizarse.

EL PROGRAMA

El programa utiliza la totalidad de los gráficos de usuario, de los cuales veinte definen la calavera

La subrutina generadora de caracteres gigantes, amplía el patrón habitual de un carácter del Spectrum, tanto en anchura como en altura. En el gráfico se muestra una ampliación de 3×2.



¡!

Los caracteres que aparecen con subrayado simple en el listado, corresponden a los gráficos de las teclas cuyo carácter se subraya.

*

Los gráficos editados junto con **CAPS SHIFT**, se representan mediante la tecla correspondiente doblemente subrayada.

¡!

Para la necesaria carga del código máquina, es imprescindible que el programa sea ejecutado desde la línea 1650.

*

El código máquina de la subrutina de caracteres gigantes, ocupa 277 bytes a partir de la dirección 32256.

*

El diseño del programa asegura que las palabras propuestas por el Spectrum sean escogidas al azar y sin repetición, hasta que el programa no es interrumpido o cargado nuevamente.

*

El código máquina debe estar situado en la cinta inmediatamente después del listado BASIC.

que aparece inscrita en la bomba, y el restante, el brazo extensible que porta el detonador. Por tanto, conviene recordar que los caracteres que aparecen subrayados en el listado, equivalen a los gráficos de las teclas correspondientes; por otra parte, siguiendo nuestra norma en materia de listados, los caracteres doblemente subrayados (sólo podrán ser números), corresponden a los gráficos cambiados de dichas teclas, es decir, **CAPS SHIFT** + tecla (en modo gráfico).

En cuanto al fichero interno que contiene las 50 palabras que puede elegir al azar el Spectrum, puede ser ampliado siempre que en la matriz **D** definida en la línea 80, realicemos la pertinente actualización; del mismo modo, también será necesario ampliar el rango de elección del **RND** (línea 650).

Tal y como ha sido diseñado el programa, las palabras se seleccionan aleatoriamente y sin repetición. Debido a esto, puede darse el caso de que al agotarse el banco de palabras, el programa se detenga con el mensaje **E Out of DATA.** En este caso deberemos efectuar **RUN,** para reciclar todo el vocabulario.

A la hora de grabar el programa, lo haremos mediante el uso combinado de los comandos SAVE y LINE de la siguiente forma:

SAVE "BOMBA" LINE 1650

Para una mayor estética del programa, hemos utilizado la subrutina de caracteres gigantes de PSION Computers, incluida en la casete de demostración de los primeros Spectrum. Dicha subrutina deberá ser grabada a continuación del pograma BASIC con la dirección 32256 y una longitud de 277 bytes.

Para llevar a cabo esta última parte de la adopción del programa principal, introduzca en la memoria el siguiente programa:

```
10 REM CARGADOR DE CODIGO MAQUINA
20 CLEAR 32255
30 FOR I=32256 TO 32532
40 INPUT A
50 POKE I,A
60 NEXT I
70 PRINT "SITUA LA CINTA AL FINAL DEL BASIC"
80 SAVE "LIT" CODE 32256,277
```

Este programa realizará la petición de los 277 números correspondientes al listado de código máquina (introduzcámoslos por filas); una vez que haya finalizado la entrada de datos, se preparará la grabación del código máquina, para lo cual deberemos situar la cinta inmediatamente a continuación del final del BASIC anteriormente grabado (listado BASIC del programa LA BOMBA).

```
 10 REM **********************
 20 REM * J.M.MAYORAL SERRANO *
 30 REM **********************
 40 LET SW=0
 50 REM GRAFICOS DEF.
 60 LET ABDUL=257: PAPER 0: BORDER 0: CLS
 70 LET G$="ABCDFGHJKLMNOPQRSTUEI"
 80 DIM D(50)
 90 PRINT  FLASH 1; PAPER 5; INK 1;AT 10,3;"   UN MO
MENTO POR FAVOR
 100 PRINT  INK 5, PAPER 1; FLASH 1;AT 12,6;" ~GRAFIC
OS USUARIO ~"
 110 DATA BIN 00000111,BIN 00001111,BIN 00001111,BIN
00000111,BIN 01110110,BIN 11111111,BIN 11111111,BIN 0
1111011
 120 DATA BIN 0,BIN 10000000,BIN 10000000,BIN 0,BIN 0
,BIN 10000000,BIN 11000000,BIN 11100000
 130 DATA BIN 0,BIN 1,BIN 1,BIN 0,BIN 0,BIN 1,BIN 11,
BIN 111
 140 DATA BIN 11100000,BIN 11110000,BIN 11110000,BIN
11100000,BIN 01101110,BIN 11111111,BIN 11111111,BIN 1
```

```
1011110
 150 DATA BIN 11110000,BIN 11111000,BIN 01111100,BIN
00111110,BIN 00011101,BIN 00001011,BIN 00000011,BIN 1
11
 160 DATA 0,0,BIN 00111100,BIN 01111110,255,255,255,2
55
 170 DATA BIN 1111,BIN 11111,BIN 00111110,BIN 0111110
0,BIN 10111000,BIN 11010000,BIN 11000000,BIN 11100000
 180 DATA BIN 111,BIN 1111,BIN 1111,BIN 00011110,BIN
00011100,BIN 00001100,BIN 00000110,BIN 00001011
 190 DATA 255,255,255,254,BIN 011111100,BIN 00111000,
BIN 00010000,BIN 00010001
 200 DATA BIN 11100000,BIN 11110000,BIN 11110000,BIN
11111000,BIN 01111000,BIN 01110000,BIN 11100000,BIN 1
1010000
 210 DATA 0,0,0,0,BIN 00011101,BIN 001111111,BIN 0011
1111,BIN 00111111
 220 DATA BIN 00011011,BIN 00111101,BIN 01111101,BIN
11111011,BIN 11110011,BIN 11100111,BIN 11000111,BIN 1
0000011
 230 DATA BIN 10111011,255,BIN 11101111,BIN 11000111,
BIN 10000011,255,255,255
 240 DATA BIN 11011000,BIN 10111100,BIN 10111110,BIN
11011111,BIN 11001111,BIN 11100111,BIN 11100011,BIN 1
1000001
 250 DATA 0,0,0,0,BIN 10110000,BIN 1111100,BIN 111111
00,BIN 10111100
 260 DATA BIN 00011101,1,7,BIN 1111,BIN 1111,BIN 0000
0110,0,0
 270 DATA BIN 10000001,BIN 11000001,BIN 11000001,BIN
11000000,BIN 10000000,0,0,0
 280 DATA BIN 10011001,BIN 10011001,BIN 10011001,0,25
5,BIN 01111110,BIN 00111100,0
 290 DATA BIN 10000001,BIN 10000011,BIN 10000011,BIN
00000011,BIN 10000001,0,0,0
 300 DATA BIN 10111100,BIN 10000000,BIN 11100000,BIN
11100000,BIN 11000000,0,0,0
 310 DATA 0,BIN 10000001,BIN 11100111,255,255,BIN 111
00111,BIN 10000001,0
 320 DATA 127,32,74,46,77,46,32,77,65,89,79,82,65,76,
32,83,69,82,82,65,78,79
 330 POKE 23658,8
```

```
340 RESTORE 10
350 FOR N=1 TO LEN G$
360 FOR K=0 TO 7
370 READ A
380 POKE USR G$(N)+K,A
390 NEXT K
400 NEXT N
410 INK 9: CLS
420 REM DIBUJO BOMBA
430 CIRCLE 51,100,40
440 PRINT  INK 6;AT 7,4;'AB CD'
450 PRINT  INK 6;AT 8,4;' EGH '
460 PRINT  INK 6;AT 9,4;' JKL '
470 PRINT  INK 6;AT 10,4;'MNOPQ'
480 PRINT  INK 6;AT 11,4;'RSTUV'
490 PLOT 46,141: DRAW 0,10
500 DRAW 12,0: DRAW 0,-10
510 FOR X=50 TO 53
520 PLOT X,151
530 DRAW 0,23
540 NEXT X
550 FOR X=48 TO 50
560 PLOT X,142: DRAW 0,7
570 NEXT X
580 FOR R=1.3 TO 1.5 STEP .05
590 PLOT 40,65
600 DRAW 0,70,-PI/R
610 NEXT R
620 LET FALLO=0
630 IF SW=1 THEN  RETURN
640 REM ELECCION PALABRA
650 LET PAL=INT (RND*50)+1
660 IF D(PAL) THEN  GO TO 650
670 LET D(PAL)=1
680 RESTORE 1140
690 FOR N=1 TO PAL
700 READ A$
710 NEXT N
720 DIM N$(LEN A$): GO SUB 930
730 REM INTRODUCCION INTENTOS
740 PRINT  PAPER 6;AT 21,0;'          INTRODUCE LETRA
    '
750 LET K$=INKEY$
760 IF CODE K$<65 OR CODE K$>90 THEN  GO TO 750
770 BEEP .2,40
780 PRINT AT 21,0; PAPER 0;'
    '
790 REM JUEGO
800 LET SWCO=0
810 FOR N=1 TO LEN A$
820 IF A$(N)=K$ THEN  PRINT  INK 6;AT 10,14+N;K$: LE
T SWCO=1: LET N$(N)=A$(N)
830 NEXT N
840 IF A$=N$ THEN  GO TO 1010
850 IF SWCO THEN  GO TO 730
860 PRINT AT 19,(FALLO+3)*2;K$
870 LET FALLO=FALLO+1
880 RANDOMIZE
890 IF FALLO=8 THEN  GO SUB 980: GO TO 1200
900 GO SUB 980
910 GO TO 730
920 REM  RAYITAS
930 FOR R=1 TO LEN A$
940 PRINT  INK 4;AT 10,14+R;'-'
950 NEXT R
960 RETURN
970 REM ANIMACION
980 PRINT  INK 4;AT 0,31-FALLO*3;'III'
990 PRINT  OVER 1; BRIGHT 1; FLASH 1; PAPER 6; INK 2
;AT 0,30-FALLO*3;'8'
1000 RETURN
1010 REM  PALABRA ACERTADA
1020 FOR N=0 TO 31
1030 PRINT  PAPER 0;AT 19,N;' '
1040 NEXT N
1050 LET P$='MUY BIEN MUCHACHO!'
1060 LET YY=130: LET XS=1: LET YS=2
1070 BRIGHT 1: INK 4
1080 GO SUB 1550: INK 9: BRIGHT 0
1090 FOR N=1 TO 50
1100 BORDER 3:::::: BORDER 4: BORDER 6:
1110 BEEP .01,40
1120 NEXT N
1130 PAPER 0: BORDER 0: CLS : GO TO 430
1140 REM B.DATOS (50 PALABRAS)
1150 DATA 'TRANSFORMADOR','BUTACON','CARPETA','TELEFO
NO','SOFA','CIGARRO','SPECTRUM','IMPRESORA','MICROSCO
PIO','CARTERA'
1160 DATA 'LIBRO','INTERFACE','PESCADO','LEOPARDO','C
OMPAS','TELEVISOR','COCINA','LAGARTO','ZAPATILLA','TE
CNOLOGIA'
1170 DATA 'LEXICO','PTERODACTILO','LASER','CENICERO',
'CUARTILLA','GATITA','ANALISTA','FERNANDO','INGELEK',
'TELEX'
1180 DATA 'VISITANTE','VULGARIDAD','ZANAHORIA','PSIQU
IATRIA','MAYORDOMO','MAYORAL','FOTOGRABADO','MILITAR'
,'AMETRALLADORA','ESTADISTICA'
1190 DATA 'DESLIZAR','ABECEDARIO','COMBUSTIBLE','CABE
CERA','AXIOMATICO','XILOFONO','VENTRICULO','NITROGLIC
ERINA','PIRAMIDE','LEPIDOPTERO'
1200 REM  FIN JUEGO
1210 FOR N=31 TO 7 STEP -1
1220 PRINT  PAPER 0;AT 0,N;' '
1230 NEXT N
1240 PRINT  PAPER 2; INK 6; FLASH 1; BRIGHT 1; OVER 1
;AT 1,6;' ';AT 2,6;' '
1250 FOR N=1 TO 20
1260 BEEP .05,40: BEEP .04,30
1270 NEXT N
1280 FOR Y=175 TO 159 STEP -1
1290 PLOT 50,Y
1300 DRAW  FLASH 0; BRIGHT 0; OVER 1; PAPER 0; INK 0;
5,0
1310 NEXT Y
1320 FOR C=7 TO 0 STEP -1
1330 FOR N=1 TO 12
1340 PRINT  OVER 1; PAPER C; INK 9;AT N+2,1;'
    '
1350 NEXT N
1360 NEXT C
1370 FOR N=1 TO 12
1380 PRINT  PAPER 0;AT N+2,1;'            '
1390 NEXT N: CLS
1400 LET P$='BOOM!!'
1410 LET XS=5: LET YS=8: LET YY=30
1420 INK 3
1430 GO SUB 1550
1440 LET P$='LA PALABRA ES...'
1450 LET XS=2: LET YS=3: LET YY=100
1460 INK 7
1470 GO SUB 1550
1480 LET P$=A$
1490 LET XS=2: LET YS=2: LET YY=130
1500 INK 6
1510 GO SUB 1556
1520 IF INKEY$='' THEN  GO TO 1520
1530 INK 9: CLS : GO TO 430
1540 REM SBR. CARACT. GIGANTES
1550 LET xx=(256-8*xs*LEN p$)/2
1560 LET i=23306
1570 POKE i,xx: POKE i+1,yy: POKE i+2,xs: POKE i+3,ys
: POKE i+4,8
1580 LET i=i+4: LET w=LEN p$
1590 FOR q=1 TO w
1600 POKE i+q,CODE p$(q)
1610 NEXT q
1620 POKE i+w+1,255
1630 LET w=USR 32256
1640 RETURN
1650 REM  AUTO-EJEC. OBLIGATORIA
1660 CLEAR 32255
1670 LOAD ''CODE
1680 CLS
1690 LET SW=1: GO SUB 50
1700 LET SW=0
1710 LET P$='LA'
1720 LET XX=100: LET XS=2: LET YS=2: LET YY=30
1730 GO SUB 1560
1740 LET XS=4: LET YS=11: LET YY=50
1750 LET K$='BOMBA'
1760 FOR N=1 TO 5
1770 LET XX=N*32+65
1780 LET P$=K$(N)
1790 INK N+1
1800 GO SUB 1560
1810 NEXT N
1820 LET P$=''
1830 FOR N=1 TO 22
1840 READ ABDUL
1850 LET P$=P$+CHR$ ABDUL
1860 NEXT N
1870 PRINT  INK 9;AT 20,1;P$
1880 FOR N=1 TO 200
1890 NEXT N
1900 BEEP .5,30
1910 CLS
1920 INK 9
1930 GO TO 430
```

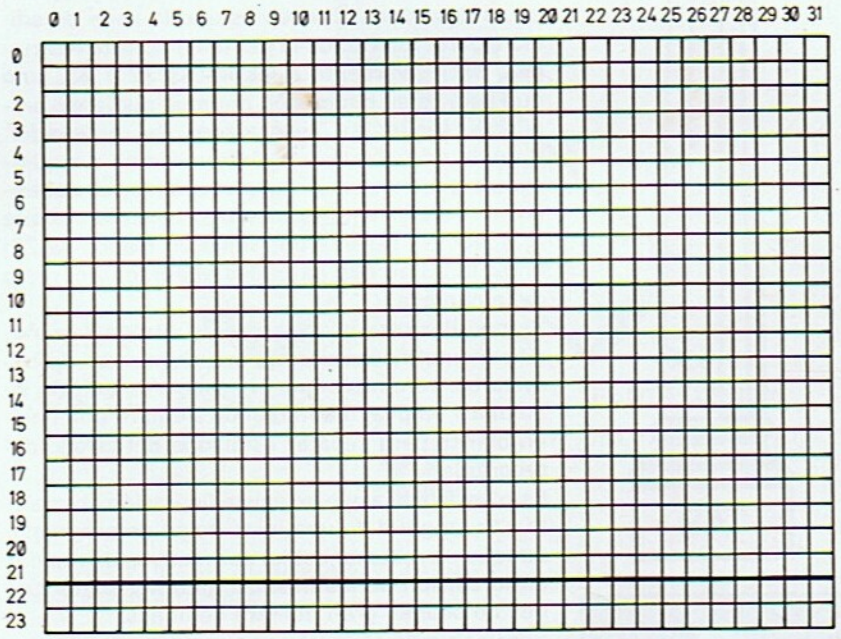
33	15	91	126	35	34	0	91
111	60	200	38	0	41	41	41
237	75	54	92	9	62	8	50
4	91	58	11	91	50	9	91
58	10	91	50	8	91	62	9
50	5	91	126	35	34	2	91
7	50	6	91	58	5	91	61
32	50	58	4	91	61	32	24
58	14	91	71	58	12	91	79
58	10	91	129	5	32	252	50
10	91	42	0	91	195	3	126
50	4	91	58	13	91	71	58
9	91	128	50	9	91	42	2
91	195	32	126	50	5	91	58
12	91	71	58	9	91	50	7
91	58	13	91	79	197	205	164
126	193	58	7	91	60	50	7
91	13	32	241	58	8	91	60
50	8	91	5	32	221	58	6
91	195	48	126	128	64	32	16
8	4	2	1	58	142	92	238
255	71	58	141	92	160	71	58
8	91	230	248	111	58	7	91
254	192	208	31	31	31	230	31
103	203	28	203	29	203	28	203
29	203	28	203	29	62	88	180
103	58	142	92	166	176	119	58
7	91	71	230	7	246	64	103
120	31	31	31	230	24	180	103
120	23	23	230	224	111	58	8
91	71	31	31	31	230	31	181
111	235	33	156	126	120	230	7
79	6	0	9	70	26	33	6
91	203	70	40	3	176	18	201
47	176	47	18	201			

SITUACION EN LA PANTALLA

NO cabe duda de que la parte más espectacular de la programación, es la impresión en la pantalla de literales y resultados; la sentencia BASIC que permite efectuar estas salidas al «exterior», de los contenidos de la memoria del ordenador es **PRINT**.

Hasta ahora hemos analizado la sentencia **PRINT** por sí misma, y en combinación con algunos signos de puntuación. Podemos hacer aquí un breve repaso de todo lo dicho, y ampliar conocimientos con nuevas posibilidades de **PRINT**, la más versátil de las palabras BASIC.

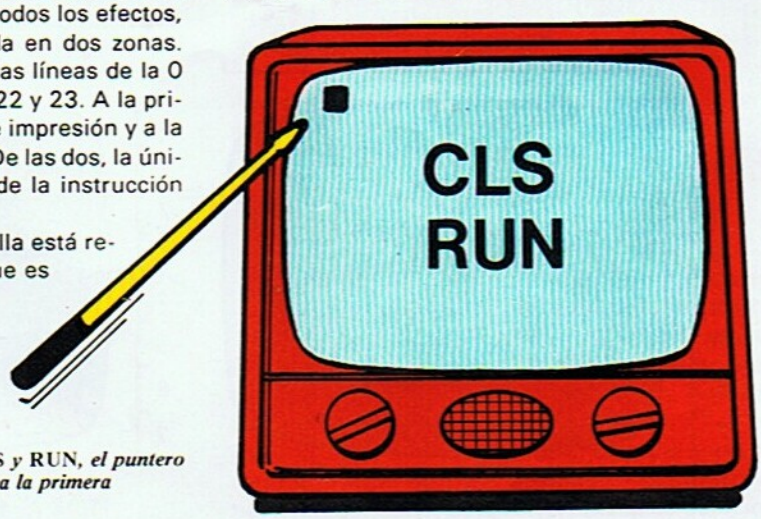
LA ESTRUCTURA DE LA PANTALLA



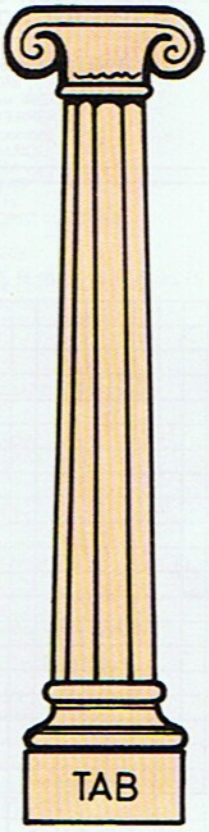
La pantalla del Spectrum está compuesta por 24 filas de 32 columnas, de las cuales las dos últimas filas están habitualmente reservadas para el uso del sistema.

La pantalla del Spectrum consta de 24 líneas de 32 columnas, numeradas simbólicamente de la 0 a la 23 y de la 0 a la 31, respectivamente, cuyo origen se encuentra en el ángulo superior izquierdo de la pantalla. De esta forma, es posible imaginarnos la pantalla del ordenador como una hoja de papel cuadriculado de dimensiones 24×32, sobre la que podemos escribir y reescribir a nuestro gusto, como si de una pizarra se tratase. Debemos saber también que, a todos los efectos, la pantalla se encuentra dividida en dos zonas. La primera de ellas comprende las líneas de la 0 a la 21, y la segunda las líneas 22 y 23. A la primera zona la denominaremos de impresión y a la segunda, zona del «ordenador». De las dos, la única que es accesible por medio de la instrucción **PRINT** es la primera.

El segundo apartado de la pantalla está reservado al ordenador, puesto que es la zona que se emplea para la edición de las instrucciones BASIC, para su introducción, modificación o borrado. Son también las líneas encar-



Cuando se ejecutan los comandos CLS *y* RUN, *el puntero de pantalla (posición de* PRINT*) indica la primera posición de la misma.*



La función BASIC TAB *se utiliza para el desplazamiento de la posición de impresión por columnas.*

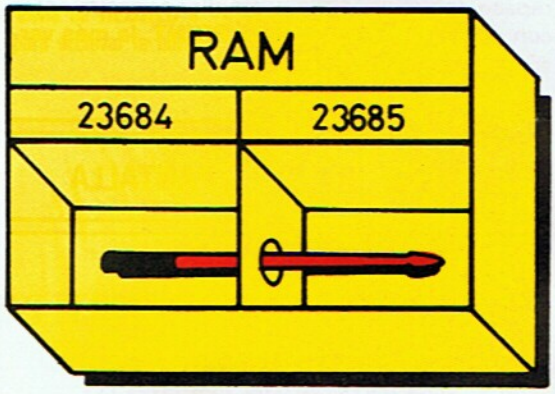
gadas de recoger los mensajes del Spectrum, tanto de advertencia como de error.
Si hemos sido observadores durante la introducción de los programas de ejemplo, nos habremos dado cuenta de la existencia bien diferenciada de estas dos zonas. El caso es que la zona de edición (parte inferior de la pantalla) no está restringida exclusivamente a las líneas 22 y 23, sino que ésta es su dimensión mínima, pudiendo ampliarse según las necesidades del ordenador. Cuando en un programa introducimos o modificamos una instrucción que ocupa menos del ancho de una línea (32 caracteres), observamos que el Spectrum reserva una línea en blanco (la 22) entre la 23 en que parpadea el cursor y el resto de la pantalla.
Si, por el contrario, tratamos de introducir o modificar una instrucción de mayor número de caracteres, veremos cómo el espacio de edición se amplía y amplía, manteniendo siempre una línea en blanco para separar esta zona del listado del programa.
Esto es debido a que no existe limitación en cuanto a la longitud de una instrucción de programa, aunque es recomendable, por motivos de rapidez en la edición de las mismas, que las instrucciones no ocupen gran número de líneas.
Veremos más claro el concepto reticular (cartesiano) de la pantalla por medio de un ejemplo:

```
10 REM - ESTRUCTURA DE LA PANTALLA
20 FOR I=0 TO 31
30 PRINT (STR$ I)(LEN STR$ I);
40 NEXT I
50 FOR I=1 TO 21
60 PRINT (STR$ I)(LEN STR$ I)
70 NEXT I
80 STOP
```

Por medio de este programa hemos podido ver, claramente diferenciadas, las dos zonas en que se divide la pantalla. La zona de impresión convencional y la reservada a los mensajes de error y advertencia del Spectrum. En nuestro caso, una vez ejecutado el programa, observamos en la línea 23 el mensaje:

9 STOP statement, 80:1

En la primera línea de pantalla aparecen los dígitos de las unidades de los posibles valores de las columnas. Del mismo modo, en la primera columna de todas las filas, aparecen los correspondientes a los números de línea. Estos son los márgenes en que podemos trabajar por medio de la sentencia **PRINT**.

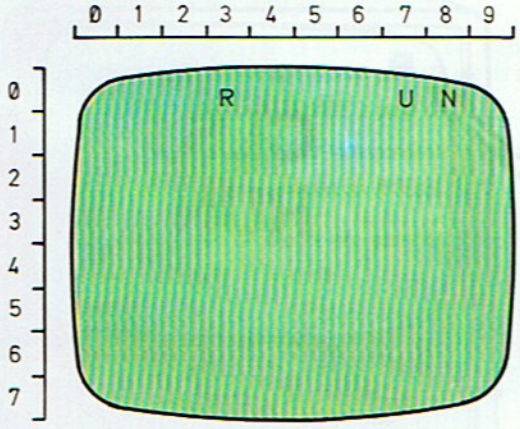


Las posiciones de memoria 23684 y 23685 (dentro de las Variables del Sistema), albergan el puntero de la pantalla.

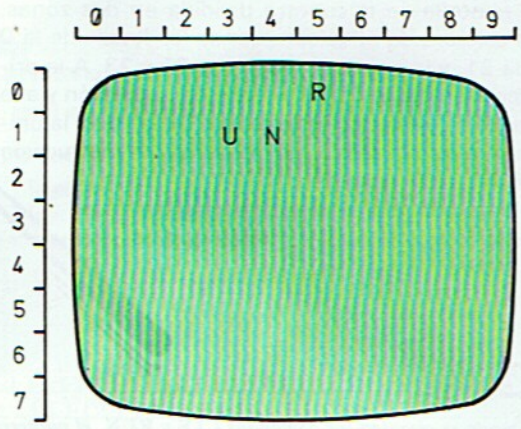
Existe una Variable del Sistema, que le indica a la sentencia **PRINT** a qué posición de la pantalla debe efectuarse la siguiente salida de un carácter. Este puntero de impresión es manejado directamente por el ordenador y, por lo tanto, no debemos preocuparnos de él.

Cuando efectuamos un borrado de pantalla por medio de las sentencias **CLS** o **RUN**, este puntero señala a la primera dirección lógica de la pantalla, que es la columna 1 de la línea 1.

Los argumentos para una serie de funciones TAB, *siempre deben estar en orden creciente, lo cual no es imprescindible utilizando* AT.



PRINT AT 0,3;"R";TAB 7;"UN"



PRINT AT 0,5;"R";TAB 3;"UN"

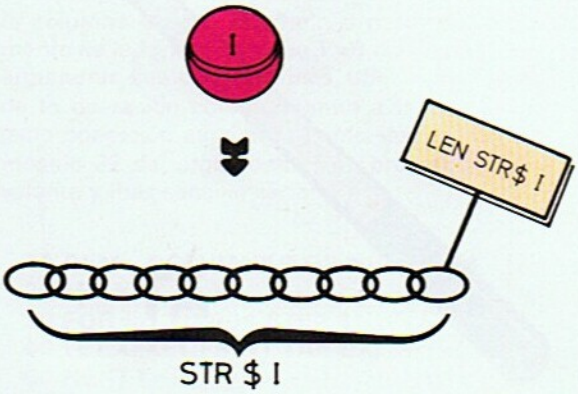
PUNTERO DE IMPRESION

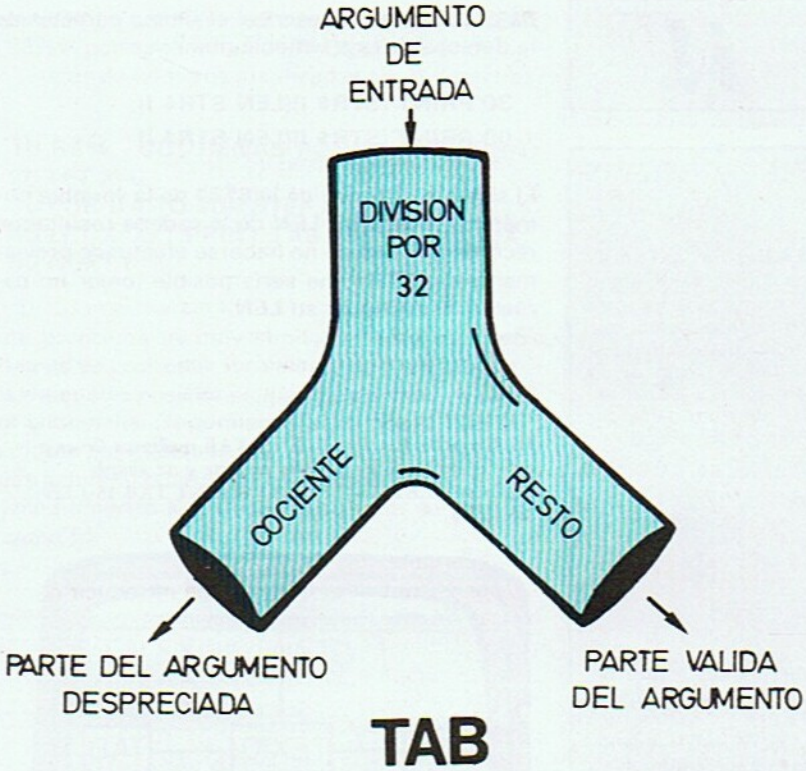
Existe una Variable del Sistema, que le indica a la sentencia **PRINT** en qué posición de la pantalla debe efectuarse la siguiente salida de un carácter. Este puntero de impresión es manejado directamente por el ordenador y, por lo tanto, no debemos preocuparnos de él. Sin embargo, hay algunas cosas sobre su funcionamiento que nos interesa saber: cuando efectuamos un borrado de pantalla por medio de las sentencias **CLS** o **RUN**, este puntero señala a la primera dirección lógica de la pantalla, que es la columna 0 de la línea 0. De esta forma, el contenido de la siguiente instrucción con un **PRINT** vendrá a parar a este lugar concreto de la pantalla.

En el caso del programa ejemplo anterior, al ejecutar **RUN**, el puntero se coloca señalando a la primera dirección lógica de la pantalla. Como a continuación existe un bucle de 32 impresiones de un carácter con punto y coma (;), cada una de ellas se coloca inmediatamente a la derecha de la anterior, sin efectuarse ningún retorno de carro, hasta completar la línea. Esto sucede en las líneas 20 a 40.

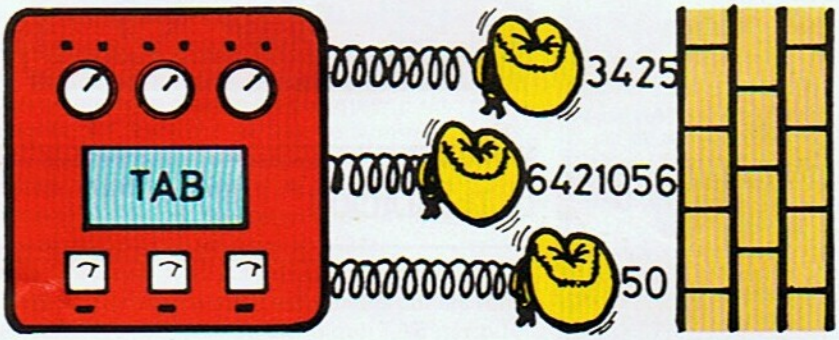
Observamos que las líneas 30 y 60 son extraordinariamente parecidas, difiriendo tan sólo en que la última no termina con punto y coma (;). Sin embargo, los resultados son completamente diferentes, puesto que en la instrucción 60 los caracteres son impresos uno a continuación de otro, pero en columna y no en la misma fila.

Para tomar el último dígito de un valor numérico, primeramente es necesario convertirlo en cadena (STR$), y posteriormente hallar el carácter correspondiente a su longitud (LEN).





La función TAB *opera con el módulo 32 del valor absoluto de su argumento.*



Apoyándonos en la función TAB, *podemos llevar a cabo la edición de valores numéricos columnados por la derecha.*

Esto es debido a que al no especificarse el punto y coma (;) al final del **PRINT** de dicha instrucción, el BASIC supone que se desea un retorno de carro, es decir, una puesta del puntero de impresión al valor de la primera columna de la siguiente línea de la pantalla.

Del programa anterior, merecen especial comentario las líneas 30 y 60; además de por lo dicho hasta el momento, porque en ambas se solucio-

¡!

TAB puede tomar cualquier valor como argumento, ya que considera como número de columna a saltar, el resto de la división por 32 del valor absoluto del argumento especificado.

*

Cuando se codifican varias expresiones **TAB** en una instrucción **PRINT**, los valores de los argumentos deben aparecer en secuencia ascendente.

*

La sentencia **TAB** permite «saltar» a un determinado número de columna, especificado como argumento de la función.

*

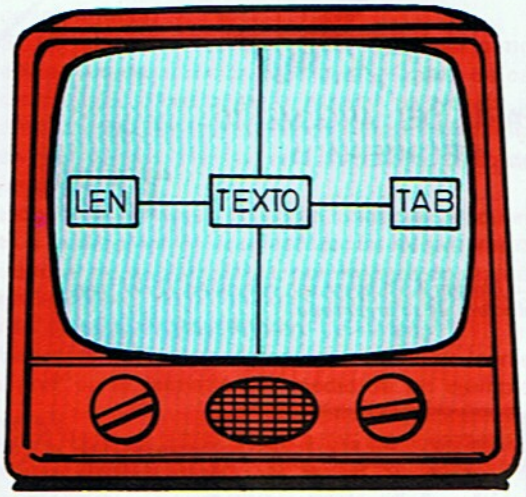
Cuando se especifica como argumento de **TAB** un número de columna ya sobrepasado, se genera un cambio automático a la siguiente línea de la pantalla, antes de saltar a la columna especificada por el argumento.

na el problema de escribir el último carácter de la derecha de una variable numérica:

```
30 PRINT (STR$ I)(LEN STR$ I);
60 PRINT (STR$ I)(LEN STR$ I)
```

El sistema es tomar de la STR$ de la variable numérica, el carácter **LEN** de la cadena resultante; recordemos que de no haberse efectuado previamente un **STR$**, no sería posible tomar un carácter, ni averiguar su **LEN**.

Mediante las funciones LEN *y* TAB *podemos llevar a cabo el centrado de literales en base a un simple algoritmo:* LET A$="TEXTO":PRINT TAB 15-LEN A$/2;A$.



LA TABULACION

El empleo de la sentencia **PRINT** con coma (,) lo hemos visto en capítulos anteriores. Como recordatorio diremos que propicia un salto al comienzo de la siguiente «mitad» de línea, suponiendo que ésta se divide simbólicamente en dos mitades de 16 columnas cada una.

La sentencia **TAB** es mucho más precisa, permitiendo «saltar» a un determinado número de columna, especificado como argumento de la función. **TAB** toma el valor absoluto de los argumentos negativos, y puede utilizar expresiones mayores que 31. Esto es posible gracias a que opera realmente con el resto de la división entre 32 del argumento. De esta forma, las expresiones siguientes producen el mismo efecto: **PRINT TAB 5, PRINT TAB 37.**

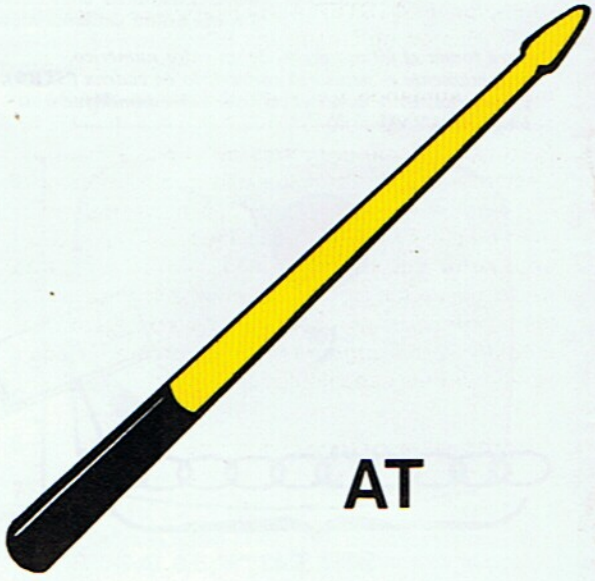
Una restricción en cuanto a los argumentos de la función **TAB** es que éstos, en el caso de existir más de uno, deben estar ordenados en forma ascendente, es decir, el **TAB** nunca da «marcha atrás». Un ejemplo correcto sería: **PRINT TAB 5;1;TAB 10;2;TAB 15;3.** También lo sería, si nos fijamos en el tratamiento que la función **TAB** da a sus argumentos, haber escrito 47 en lugar de 15, en la última tabulación de la línea del ejemplo.

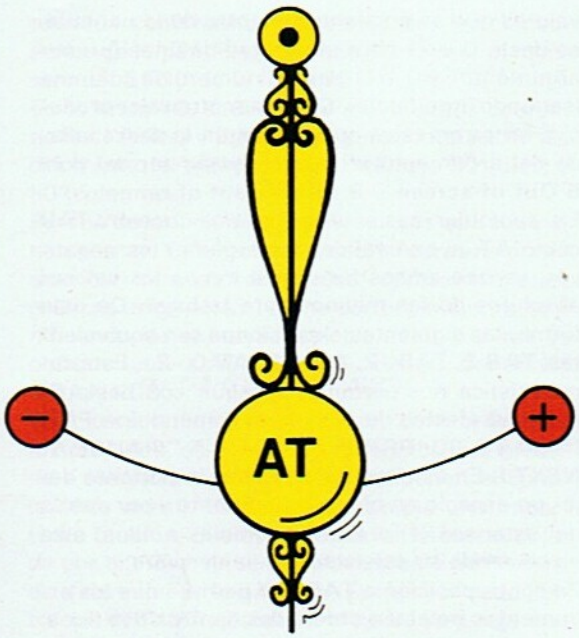
Si escribimos: **PRINT TAB 10;1;TAB 5;2;TAB 15;3** los resultados que obtenemos son diferentes. Esto es debido a que, como ya hemos dicho, no podemos especificar como argumento de la función **TAB** un número inferior al de la columna de la última tabulación realizada.

En la línea del ejemplo, la primera tabulación se realiza correctamente a la columna 10, pero la segunda, al no poder efectuarse en esa línea un salto a una columna anterior, genera automáticamente un cambio a la siguiente línea, y después efectúa la tabulación a la columna 5. El **TAB** a la columna 15 se realiza correctamente por ser esta columna mayor que 5, pero se efectúa en la misma línea que su tabulación precedente.

TAB nos sirve para imprimir a partir de una columna concreta de la pantalla, pero nos puede brindar posibilidades más concretas. Por ejemplo, obtener columnas de números en el lugar deseado:

La función AT *sitúa el puntero de pantalla en cualquier posición de la misma que nosotros indiquemos.*





Gracias a que la función AT *actúa sobre el valor absoluto de sus parámetros, podemos apoyarnos en ella para la programación de móviles con desplazamiento pendular.*

```
10 REM - COLUMNAS DE NUMEROS
20 LET X=0
30 FOR I=0 TO 21
40 FOR J=0 TO 28 STEP 4
50 LET X=X+1:PRINT TAB J;X;
60 NEXT J
70 NEXT I
```

En este programa hemos hecho un recorrido por las líneas de pantalla de la 0 a la 21, imprimiendo en cada una de ellas sólo en las columnas múltiplos de 4 **(STEP 4)**, partiendo de la columna 0. Hemos efectuado un **PRINT** con punto y coma (;), para no generar cambios de línea intermedios, sumando 1 a la variable contadora X por cada nueva impresión.

Pero este mismo programa se puede simplificar haciendo uso de la forma con que **TAB** calcula la columna en que imprimir, a partir del argumento de la función; ya que **TAB** divide por 32 el argumento que especificamos, utilizando el resto de la operación como columna a la que saltar (esta operación se conoce como «extracción del módulo 32 del argumento»); un programa equivalente y más sencillo sería:

```
10 REM - COLUMNAS DE NUMEROS
20 LET X=0
30 FOR I=1 TO 704 STEP 4
40 LET X=X+1:PRINT TAB I;X;
50 NEXT I
```

También, en combinación con las funciones **LEN** y **STR$**, podemos emplear **TAB** para obtener las columnas de números justificadas por la derecha:

```
10 REM - COLUMNAS POR LA DERECHA
20 LET X=0
30 FOR I=4 TO 704 STEP 4
40 LET X=X+1:PRINT TAB I-LEN STR$ X;X;
50 NEXT I
```

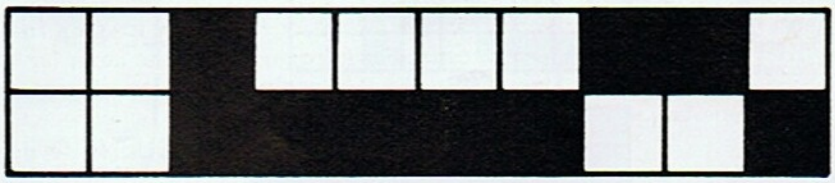
Este programa es muy similar al anterior, pero además de presentar los valores de X en intervalos de cuatro posiciones, justifica dichos valores por la derecha. De comienzo, situamos el **TAB** en la última posición del intervalo, para restarle después la longitud del valor *string* (cadena) de la variable numérica X, e imprimir su valor con punto y coma (;).

IMPRESION EN UN PUNTO DETERMINADO

Del mismo modo que **TAB** nos permite saltar a una columna concreta, dentro de la línea actual sobre la que se está produciendo la impresión, existe otra función, más completa, que permite el posicionamiento directo en cualquier punto de la pantalla: **AT**. Esta vez, se han de suministrar dos valores separados por comas, de los cuales, el primero indica el número de línea (0 a 21), y el segundo el número de columna de la línea especificada en el valor anterior (0 a 31).

Debemos hacer ahora algunas matizaciones referentes a los argumentos empleados por las funciones **TAB** y **AT**. Hemos dicho que **TAB** permite argumentos mayores que 31, al tomar el resto de la división por 32 del argumento especificado, como número de columna a saltar; pero no es este el caso de **AT**. La función **AT** sólo permite

Sprite *utilizado para el programa «ATERRIZAJE».*



¡!

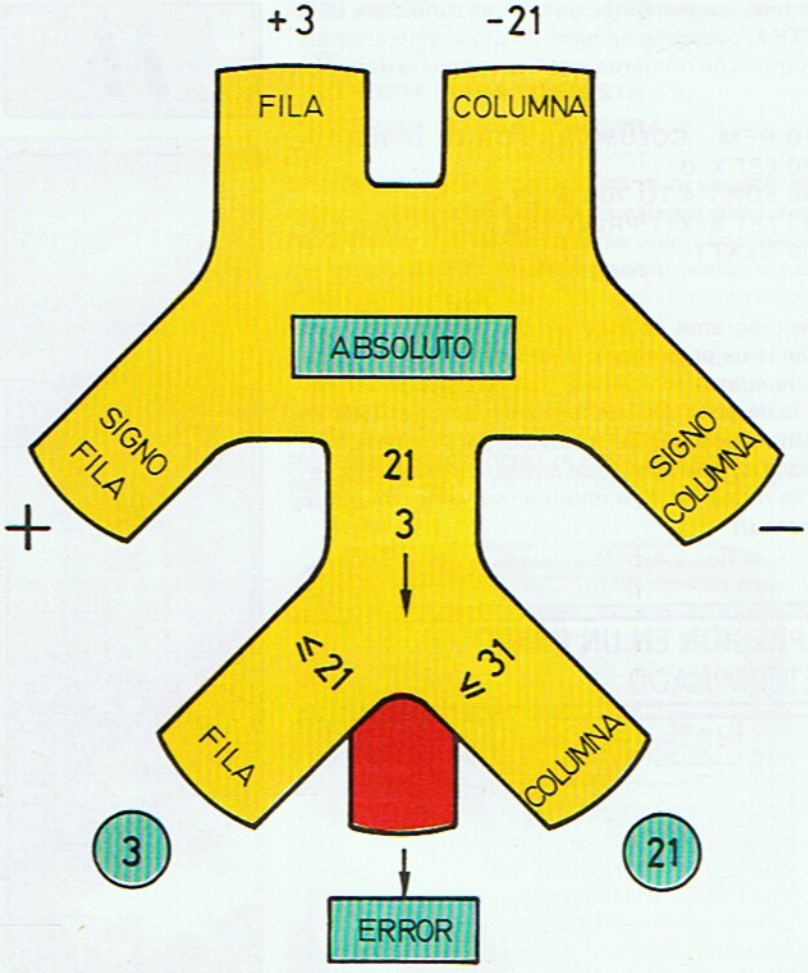
En combinación con las funciones **LET** y **STR$**, podemos emplear **TAB** para obtener columnas de números justificadas por la derecha.

*

La función **AT**, permite un posicionamiento directo en la línea y columna especificadas en sus dos argumentos, separados por una coma.

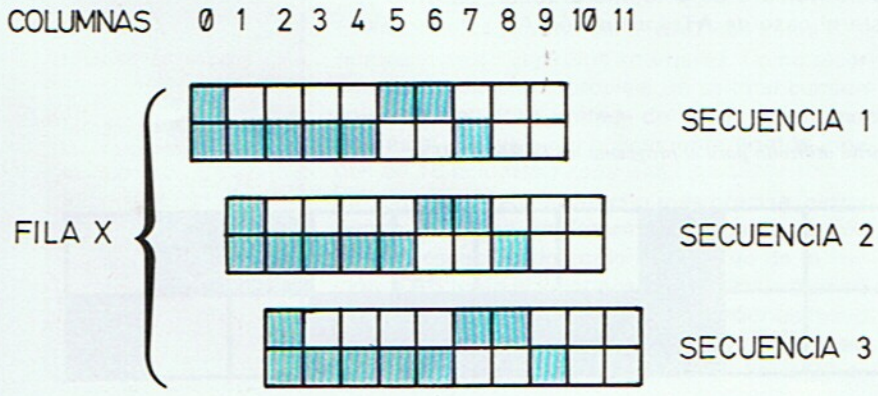
*

La función **AT** sólo permite valores que se encuentren dentro de la pantalla, es decir, 0 a 21 para el número de línea (primer argumento), y 0 a 31 para el número de columna (segundo argumento).



En el proceso de ejecución de la función AT, primeramente se realiza un acondicionamiento de los parámetros (hallando su valor absoluto), para acto seguido llevar a cabo una depuración de los valores admisibles.

En el desplazamiento de móviles a posiciones consecutivas de una misma línea de pantalla, es necesario dejar un espacio en blanco en su final, para que automáticamente la propia impresión realice el borrado de la estela.



valores que se encuentren dentro de la pantalla, es decir, 0 a 21 para el número de línea (primer argumento), y 0 a 31 para el número de columna (segundo argumento). Cualquier otro valor producirá un error; estos errores, según lo descabellado del argumento utilizado, podrán ser del tipo: **5 Out of screen** o **B Integer out of range.**

En cualquier caso, tanto para la función **TAB** como **AT**, sí son válidos los argumentos negativos, ya que ambas funciones toman los valores absolutos de los mismos para trabajar. De esta forma, las siguientes expresiones son equivalentes: **TAB 2, TAB -2, AT 0,2, AT 0,-2**... Esta característica nos permite conseguir con bastante facilidad efectos de «va y ven» o «péndulo»: **FOR I=-30 TO 30: PRINT AT 10,I;" * ":PAUSE 1: NEXT I.** En este ejemplo, es muy importante dejar un espacio en blanco por delante y por detrás del asterisco (*) a escribir; gracias a ellos, éste va borrando su estela según se desplaza.

En contraposición a **TAB, AT** permite que los argumentos no estén ordenados dentro de la línea, es decir, podemos escribir a cualquier línea y columna, justamente en el orden deseado, y sin preocuparnos de que los argumentos vayan en orden creciente, como sucede en el caso de **TAB**, ya que los resultados finales serán idénticos.

Antes de continuar hablando de **AT**, haremos un pequeño alto en el camino para aclarar el comportamiento exacto de la función **TAB** y el carácter coma (,). Hasta ahora, para simplificar en lo posible las explicaciones sobre estos elementos de edición, hemos dicho que tanto **TAB** como coma **(,)** producen un salto en su misma línea (o a la siguiente, si precisan retroceder en columnas).

De forma similar, **AT** puede saltar a cualquier punto de la pantalla. Pero existe una diferencia fundamental en la forma de desplazarse de la última función con respecto a las primeras: mientras que **AT** simplemente desplaza la posición de impresión, tanto **TAB** como coma **(,)** escriben espacios hasta situarse en su lugar.

Esto quiere decir que si en una línea en la que se realiza un **TAB** (o una edición con coma), existiera cualquier serie de caracteres entre la posición original y la de destino, ésta quedaría inmediatamente borrada..

Una prueba evidente de las posibilidades de **AT** es la simulación de movimiento:

```
10 REM - MANIOBRA DE ATERRIZAJE
20 FOR I=0 TO 20
30 FOR J=0 TO 31
40 PRINT AT I,J;" 1366"
50 NEXT J
60 NEXT I
70 STOP
```

En nuestro ejemplo, hacemos aterrizar una tosca avioneta, que va perdiendo altura en sucesivas

pasadas, comenzando su vuelo desde la línea superior de la pantalla, para aterrizar sobre el mensaje del **STOP** de la línea 70.

En la línea 40, situamos el móvil en las coordenadas **I** (línea) y **J** (columna), siendo el recorrido del mismo el marcado en las líneas 20 (filas) y 30 (columnas).

Para la descripción de los caracteres que componen el móvil, hemos utilizado la convención de que los números subrayados representan a los gráficos asignados al dígito correspondiente, significando los subrayados doblemente los que se obtiene con **CAPS SHIFT**, dentro del modo **GRAPHICS**.

Otra característica a significar, dentro de esta misma línea, es el espacio en blanco que precede a los cuatro gráficos que componen el *sprite* (gráfico móvil). Este espacio cumple una importante función, que es la de borrar la estela dejada por el móvil, en su desplazamiento de izquierda a derecha de la pantalla, ya que por cada instrucción **PRINT** ejecutada, el *sprite* se dibuja una posición más adelante.

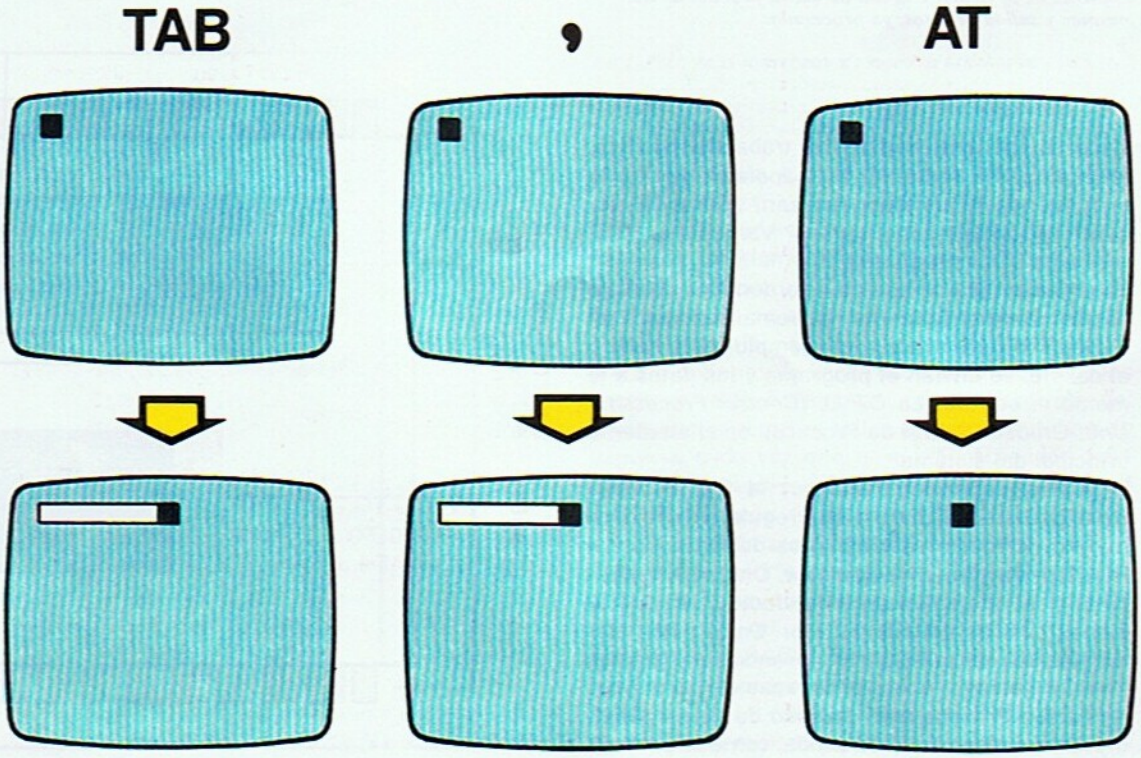
Desde luego, este programa de ejemplo podría haberse codificado por medio de la función **TAB**, ya que se produce un incremento sucesivo de la posición de tabulación, aunque otros tipos de movimiento como el de despegue, sólo se puede simular con **AT**:

```
10 REM - MANIOBRA DE DESPEGUE
20 FOR I=20 TO 0 STEP -1
30 FOR J=0 TO 31
40 PRINT AT I,J;" 1366"
50 NEXT J
60 PRINT AT I+1,0;"     "
70 NEXT I
```

Para conseguir que la avioneta despegue, han sido necesarias ciertas modificaciones en el programa. La más importante, la de la instrucción 20, para que la secuencia de líneas sobre las que imprimir vaya desde abajo hacia arriba. La segunda, la introducción de la línea 60, que antes no era necesaria. La misión de la línea 60 es la de borrar la imagen de la avioneta, cada vez que desaparece por completo por el margen derecho de la pantalla, ya que al imprimir un literal de 5 caracteres de longitud en la columna 31 de una línea, los cuatro últimos (la parte de avioneta del *sprite)* pasan a imprimirse como los primeros de la línea siguiente.

Por supuesto que el empleo de **AT** no está restringido a la simulación de movimiento, sino que es la forma más rápida y fácil de imprimir en la pantalla, cuando los literales o datos no tienen por qué ir unos detrás de otros, en una misma línea.

Los elementos de edición* TAB *y coma (,), además de producir un desplazamiento de la posición de impresión, dejan una estela (rastro) de espacios en blanco a su paso.



¡!

En contraposición a **TAB**, **AT** permite que los argumentos no estén ordenados dentro de la línea, es decir, podemos escribir en cualquier línea y columna y en el orden deseado.

*

La pantalla del Spectrum consta de 24 líneas de 32 columnas, numeradas simbólicamente de la 0 a la 23, y de la 0 a la 31, respectivamente, cuyo origen se encuentra en el ángulo superior izquierdo de la pantalla.

*

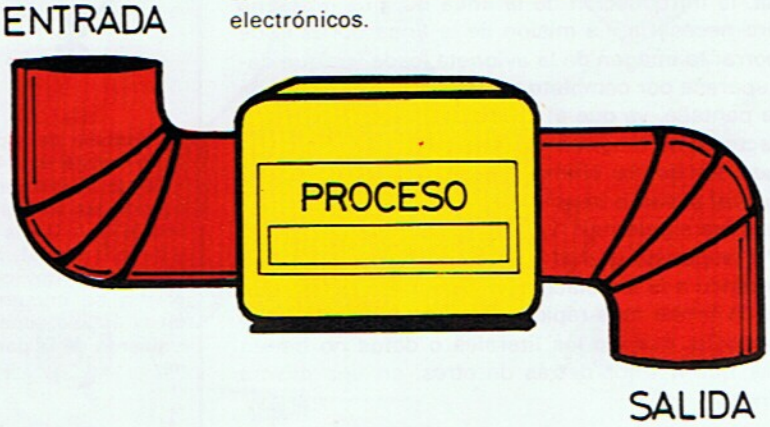
La pantalla se encuentra dividida en dos zonas. La primera de ellas comprende las líneas de la 0 a la 21, y la segunda las líneas 22 y 23. La primera zona es accesible por medio de **PRINT**, mientras que la segunda está reservada para uso del ordenador.

*

La zona inferior de la pantalla, es utilizada por el Spectrum para la edición de líneas de programa, e impresión de los mensajes de error y advertencia.

EL INTERIOR DEL SPECTRUM

CONECTAMOS a la red nuestro Spectrum. Encendemos el televisor y sobre la pantalla aparece el mensaje inicial de Sinclair. La labor es rutinaria; seguramente, la hemos realizado cientos de veces sin pararnos a pensar que, a partir de ese momento, miles de impulsos eléctricos comienzan a recorrer otros tantos diminutos circuitos electrónicos.



La organización de un ordenador responde básicamente al esquema de la figura: entrada de datos, proceso de los mismos y salida de éstos, ya procesados.

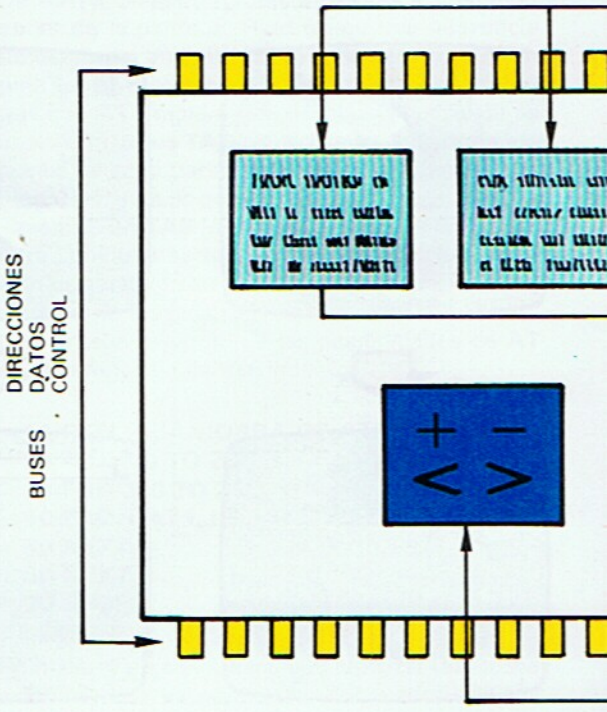
Cada componente realiza un trabajo específico, encaminado a solventar la «papeleta» que se le encomienda. ¿Qué labor realizan? ¿Cómo la ejecutan? ¿Dónde están situados? Vamos a contestar todas estas preguntas.

La organización de cualquier ordenador obedece básicamente al siguiente esquema: mediante un dispositivo de entrada, por ejemplo, el teclado o el casete, se envían el programa y los datos a la memoria central. La C.P.U. *(Central Processing Unit,* Unidad Central de Proceso), es el elemento principal del sistema.

La C.P.U. está compuesta por la C.U. *(Control Unit,* Unidad de Control), que regula el funcionamiento de todos los elementos del ordenador y la A.L.U. *(Arithmetic-Logic Unit,* Unidad Aritmético-Lógica) que se encarga de efectuar las operaciones básicas del computador. Entre estos elementos se realizan las transferencias principales de información, y sobre ellos recae el mayor peso en cualquier trabajo de proceso de datos. Finalmente, un dispositivo de salida, como la pantalla o la impresora, muestra los resultados obtenidos. Pero, entre la memoria y los dispositivos de entrada/salida, otros muchos componentes intervienen en la preparación de la información que ha de ser enviada o recibida. Vamos a ver dónde están albergados cada uno, y las funciones que desempeñan.

EL MICROPROCESADOR

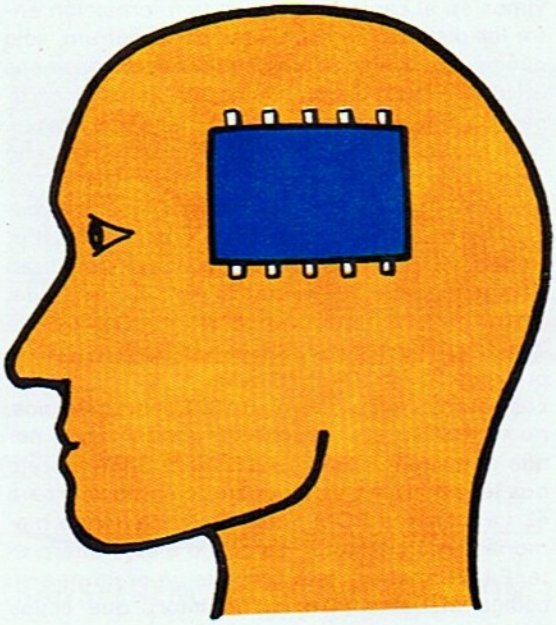
El cerebro de nuestro Spectrum es el microprocesador Z-80 A. Este es una versión más rápida del popular Z-80, fabricado por la firma ZILOG. Se trata de un conjunto de cerca de 8000 transistores integrados dentro de una pastilla (chip) de color negro, de la que sobresalen 40 patillas



metálicas. En su interior se encuentran: la C.P.U., con la Unidad de Control y la Unidad Aritmético-Lógica; el registro de instrucciones, que la C.P.U. utiliza para anotar la instrucción que corresponde ejecutar; el contador de programa, que recuerda dónde está la siguiente celdilla de memoria donde recoger la próxima instrucción a ejecutar y, finalmente, los 24 registros libres para el usuario.

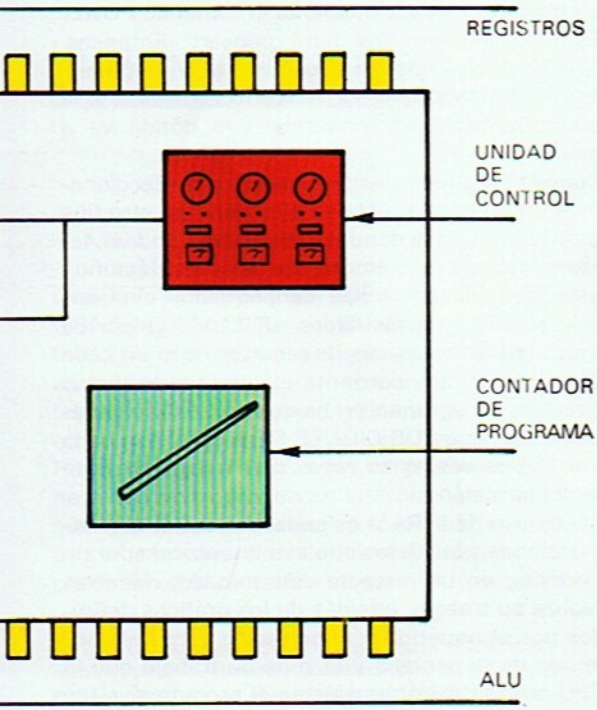
No entraremos en más detalles sobre los registros, que serán abordados adecuadamente en los capítulos dedicados al código máquina dentro de la sección BASIC.

La C.P.U., mediante su juego de instrucciones, puede realizar tareas como la lectura de datos desde un dispositivo de entrada, enviar datos hacia un dispositivo de salida, leer o escribir datos en la memoria del sistema, y efectuar operaciones aritmético-lógicas con los datos que le lleguen. Todo ello, a base de mover bits (1 ó 0) de un lado a otro.



La C.P.U. es el «cerebro» de nuestro ordenador.

Los componentes fundamentales del microprocesador Z-80 son: Unidad de Control, Unidad Aritmético-Lógica, Contador de Programa y Registros.



Todas estas operaciones, no podrían llevarse a cabo sin un elemento que fuera indicando a la Unidad de Control cuándo debe ejecutarlas, de forma que no se arme un fenomenal lío. Un circuito electrónico, que en el caso del Spectrum vibra 3,5 millones de veces cada segundo, hace las veces de reloj. Es algo así como el corazón del Spectrum; sincroniza todos los pasos que han de seguirse cada vez que el microprocesador debe ejecutar cualquier acción. De esta manera, el Z-80 A es capaz de realizar 875.000 instrucciones de las más sencillas cada segundo.

Fabuloso..., podríamos pensar. Sin embargo, no debemos creer que estas instrucciones son las mismas que utilizamos cuando escribimos un programa en BASIC. Entonces, ¿por qué se ejecuta nuestro programa, si el microprocesador no lo entiende? La razón se encuentra en otro chip, denominado memoria ROM.

¡!

En el proceso de emisión de imagen, los componentes del Spectrum que intervienen son:
* U.L.A.: Toma la imagen de la zona de la memoria R.A.M. destinada a su almacenaje, y la envía al codificador P.A.L.
* P.A.L.: Convierte las señales eléctricas recibidas de la U.L.A. en una señal de vídeo con destino al modulador.
* Modulador: Modula la señal de vídeo recibida del P.A.L., y la envía por el canal 36 de U.H.F.

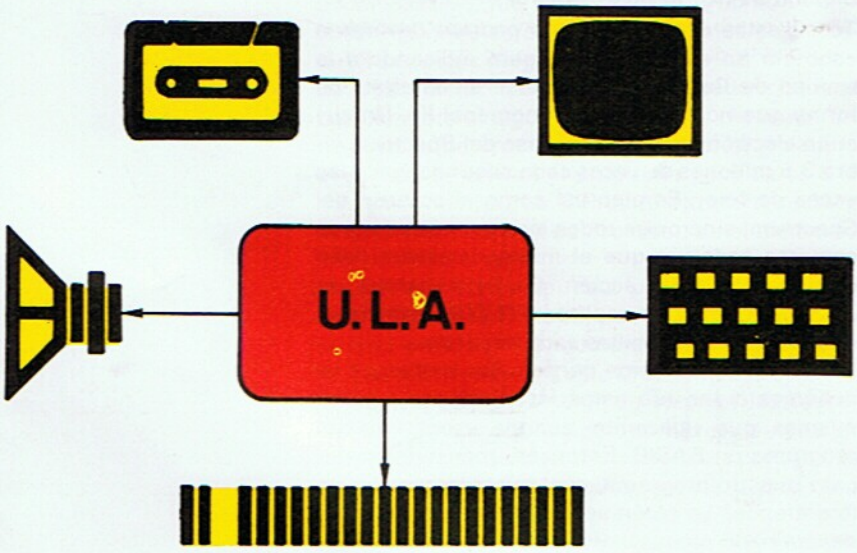
LAS MEMORIAS

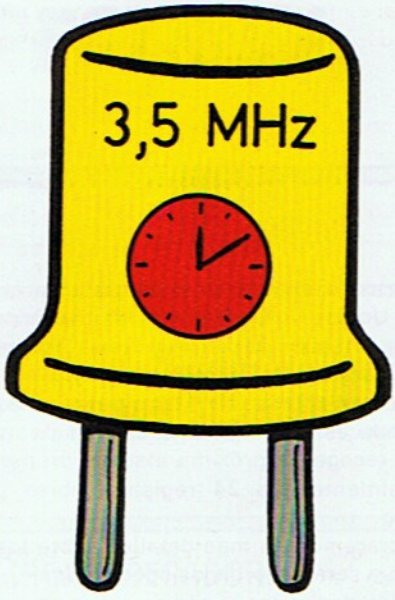
Antes de continuar, conviene recordar lo que ya vimos en el capítulo anterior: la información entre los distintos componentes del Spectrum, sólo puede estar formada por distintas combinaciones de unos y ceros denominadas bits. Estos se agrupan en unidades mayores que llamamos bytes, cada uno de los cuales está compuesto por ocho bits.

El Byte es la unidad empleada para medir la capacidad de las memorias de los ordenadores. Por tratarse de una unidad muy pequeña, se utilizan frecuentemente dos múltiplos del Byte: el KByte, KiloByte o simplemente K, equivalente a 2↑10=1024 Bytes y el MByte o MegaByte que representa 2↑20=1048576 bytes.

Los programas escritos en BASIC, como sabemos, no están formados a base de unos y ceros, por ello el microprocesador necesitará un intérprete que los traduzca a un lenguaje comprensible para él. La memoria ROM *(Read Only Memory* o memoria de sólo lectura), da cobijo a este intérprete. En su interior, lleva grabado un programa en código máquina (llamado monitor), que ocupa 16K Bytes de información (16384 bytes), desde la dirección 0 hasta la 16383.

La U.L.A. relaciona todos los periféricos con la C.P.U.





El reloj interno del Spectrum emite tres millones y medio de impulsos por segundo, marcando la cadencia de funcionamiento del ordenador.

De ellos, 7K se dedican al sistema operativo, 8K al intérprete de BASIC, y la K restante al generador de caracteres. En su interior, van incluidas las rutinas de entrada/salida, el control de la sintaxis en las expresiones BASIC, las rutinas de altavoz y teclado, etc.

Su contenido es inalterable; por más que nos empeñemos en conectar y desconectar la alimentación, o en intentar modificar las direcciones de memoria que ocupa mediante el comando **POKE**. Esta afirmación nos hará pensar: «Entonces, cuando tecleamos un programa o lo cargamos desde el casete, microdrive u otro periférico, si la memoria ROM es inalterable, ¿a dónde va a parar?»

Cuando cargamos desde el casete o confeccionamos un programa, éste se almacena en otro tipo de memoria. Es la denominada RAM *(Random Access Memory* o memoria de acceso aleatorio); está formada por miles de pequeños circuitos electrónicos, encapsulados en varios chips. Lo único que son capaces de recordar es si en cada uno de ellos hay corriente eléctrica o no. Estos circuitos se agrupan en bloques de ocho, llamados celdillas o POSICIONES DE MEMORIA, cada uno de los cuales es capaz de albergar un byte de información.

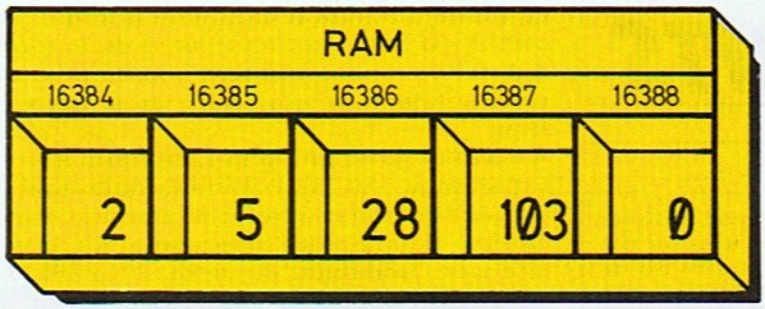
El objetivo de la RAM es poder mantener las instrucciones y los datos que el microprocesador no necesita, en un instante determinado, mientras realiza su trabajo, además de los gráficos definidos por el usuario, la información sobre el contenido de la pantalla y la zona de trabajo que la CPU precisa mientras ejecuta el programa.

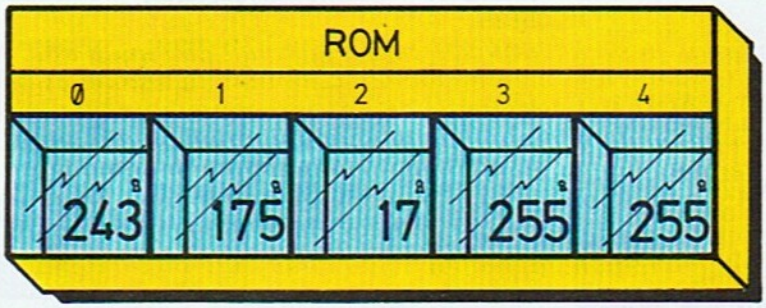
Cada celdilla dentro de la memoria lleva asignada una dirección, con objeto de identificar en cada momento su contenido. Es algo así como un enorme casillero, con un byte dentro de cada casilla. El primer byte de la memoria RAM se encuentra en la dirección 16384, mientras que el último dependerá del modelo de Spectrum (32767 en el de 16K, y 65535 en el de 48K). Las primeras 16K de memoria RAM están formadas por 8 chips, con 2K de memoria cada uno. En el Spectrum de 48K, las 32K adicionales van integradas en otros 8 chips, pero estos pueden almacenar, cada uno, 4K de información.

Incluso en el caso de tener conectado nuestro ordenador a la alimentación, el contenido de cada celdilla pierde intensidad con el paso del tiempo, debido a fugas de corriente internas. Así pues, debe ser sometido a un proceso de «refresco»; el Spectrum tiene prevista tal eventualidad y cada cierto período de tiempo, nunca superior a dos milisegundos, el microprocesador accede a cada dirección de memoria leyendo su contenido y volviéndolo a escribir.

Se trata de memorias volátiles, es decir, en el momento que cortemos la alimentación, se pierde lo que allí hubiera almacenado, al contrario de lo que ocurre en la ROM, cuyo contenido permanece inalterable.

Resumiendo, el Z-80 ejecuta las instrucciones que le van marcando los programas almacenados en las memorias RAM y ROM. La ROM se encarga de interpretar las líneas escritas en BASIC (presentes en la RAM), liberando una serie de órdenes comprensibles por el microprocesador, de manera que éste pueda realizar su trabajo.





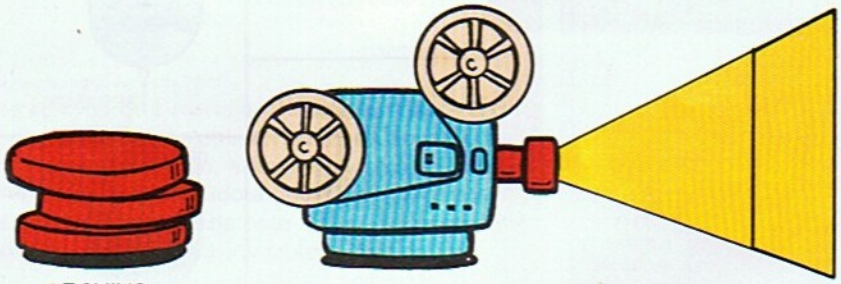
En las posiciones de memoria ROM podemos leer datos, pero no escribirlos; al contrario que en la RAM, en la que podemos efectuar ambas operaciones.

LA U.L.A.

Uno de los componentes que justifica el pequeño tamaño y reducido costo de nuestro Spectrum es la U.L.A. *(Uncommited Logic Array)*. Este chip, viene a sustituir un gran número de componentes electrónicos más sencillos, con los que se coordinan las labores de entrada/salida de datos. Puede detener el funcionamiento del reloj de manera que no se produzcan interferencias, en caso que la C.P.U. intente acceder, simultáneamente, a la misma dirección de memoria dentro del fichero de representación visual.

TRANSFERENCIAS DE INFORMACION

La comunicación entre los distintos componentes que hemos analizado, se realiza a través de una serie de conductores denominados buses. Son los encargados de que la información circule desde y hasta los dispositivos de entrada/salida a los que se desea acceder.



En el proceso de emisión de imagen hacia el televisor intervienen tres elementos fundamentales: la U.L.A., el codificador P.A.L. y el modulador de U.H.F.

BITS

En muchas ocasiones nuestro televisor nos juega la mala pasada de desajustarse; para estos momentos, nada mejor que una buena carta de ajuste que permita afinar los colores.

```
10 FOR I=0 TO 15
20 FOR J=0 TO 21
30 PRINT AT J,I*2;
BRIGHT (INT
(I/2)=I/2); INK
(I-1)/2;"■■"
40 NEXT J: NEXT I
50 BORDER 1:
BORDER 2: BOR-
DER 3: BORDER
4: BORDER 5:
BORDER 6: BOR-
DER 5: BORDER
4: BORDER 3:
BORDER 2:
BODER 1:::::GO
TO 50
```

*

Combinando adecuadamente la sentencia **PAPER** y el separador coma (,), podemos evitar en algunas ocasiones escribir líneas en blanco mediante la sentencia **PRINT**. Por ejemplo, la instrucción **PRINT PAPER 2,**, es equivalente a **PRINT PAPER 2;"32 espacios en blanco"**, pero con una ocupación de espacio considerablemente menor.

Para la transferencia de datos entre la unidad de control y la unidad aritmético-lógica del interior de la C.P.U., el microprocesador dispone de unos canales específicos, que conforman el bus interno.

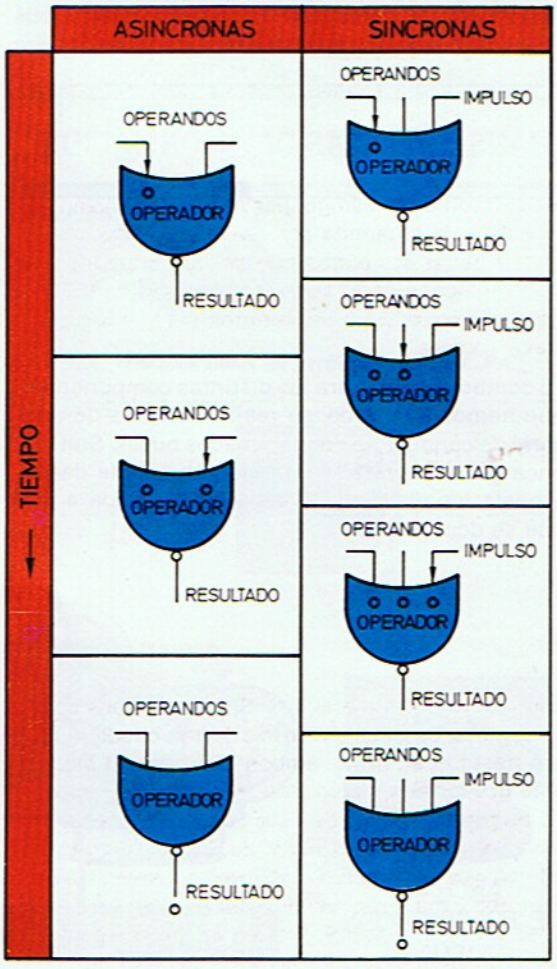
A través de las 40 patillas que conforman el encapsulado de Z-80, la C.P.U. envía o recibe señales eléctricas con un objetivo determinado. Dieciseis de estas se hallan conectadas al bus de direcciones. Tendremos, por tanto, 2↑16=65536 posibles combinaciones diferentes de ceros y unos, según pase corriente o no por cada una de ellas. Es decir, se puede acceder a cualquier dirección de memoria comprendida entre 0 y 65535.

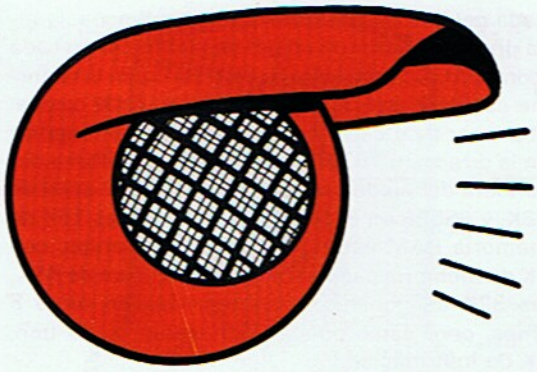
Otras ocho patillas se utilizan para enviar o recibir datos hacia la memoria o los dispositivos de entrada/salida, y están conectadas al denominado bus de datos. Como 2↑8=256, el valor de los datos que por él circulan, estará comprendido entre 0 y 255.

Cinco terminales más, reciben y transfieren señales que tienen que ver con el funcionamiento interno del microprocesador, conformando el denominado bus de control.

A los buses de datos, direcciones y control, se puede acceder directamente desde el exterior, a través de la tarjeta de expansión (port de expansión). Este elemento es el nexo de unión entre el Spectrum y los periféricos.

El ordenador realiza los procesos de manera síncrona, es decir, hace esperar la ejecución de una operación hasta que llega la señal del reloj que le da vía libre.





La emisión de sonidos en el Spectrum se efectúa a través de un pequeño zumbador (buzzer).

OTROS COMPONENTES

Las señales que se reciben o parten hacia el casete, necesitan de un interface que realice la conversión analógica/digital o digital/analógica, para transformar las señales sonoras en eléctricas binarias o viceversa. De esta manera, la C.P.U., durante el proceso de carga, realiza ciclos de lectura sucesivos, asesorada por la U.L.A., que transfiere la información al bus de datos. Es interesante señalar que en este caso el flujo de información circula en serie, es decir, bit a bit.

El modulador de color es controlado igualmente por la U.L.A., para que genere las señales correspondientes a los ocho colores de que dispone el Spectrum. A partir de él, se emite la señal de alta frecuencia que finalmente recibe la antena del televisor.

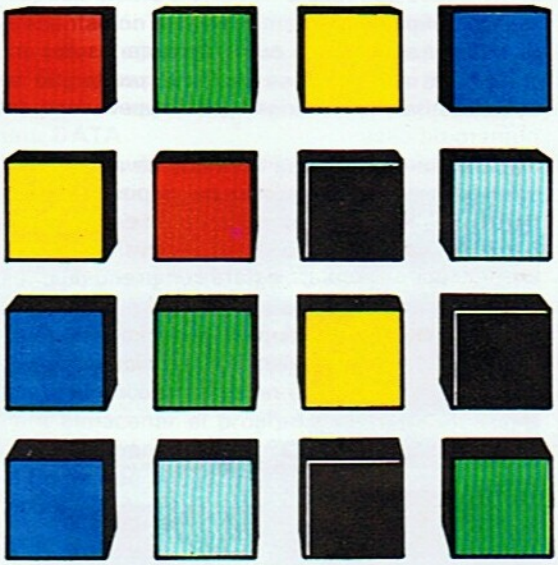
Por último, cabe mencionar la fuente de alimentación, que se encarga de suministrar al ordenador la tensión necesaria para su correcto funcionamiento. A partir del transformador de corriente exterior, que proporciona 9 voltios, el circuito 7805, colocado en el interior del Spectrum (sobre una placa metálica disipadora de calor), regula la tensión a los niveles de operación adecuados para los componentes del ordenador.

MASTER CUBOS

CUB es nuestra versión particular para RUN del conocido juego Mastermind. Por si aún queda alguien que no conozca las reglas de este pasatiempo, cosa bastante improbable, pasaremos a exponer rápidamente las mismas.

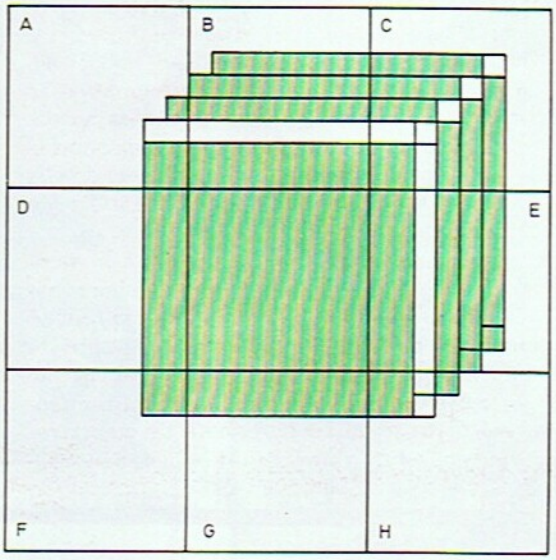
Ante todo es necesaria la presencia de dos jugadores. Uno de ellos pensará una secuencia de cuatro colores, con la condición de no repetir ninguno de ellos. En el caso concreto de CUB, este primer jugador será nuestro fiel compañero: el Spectrum.

El jugador restante, deberá adivinar dicha secuencia de colores, ejecutando combinaciones de cuatro elementos, que pueden ser o no repetidos. Lógicamente, en CUB este segundo participante seremos nosotros mismos. Para nuestras tareas detectivescas contamos con dos indispensables ayudas: por una parte, conocemos los diferentes colores con los que se ha podido formar el código secreto (siete colores), y por otra, por cada vez que realizamos un intento de descubrir el código, el Spectrum nos facilita una pista sobre lo cerca o lejos que estamos de la solución correcta. Estas pistas consisten en indicar qué colores están bien colocados en nuestra proposición, y cuáles existen en el código, pero en una posición diferente a la supuesta por nosotros. A decir verdad, y dadas las pistas que se nos brindan, nos podemos considerar buenos jugadores de CUB si no excedemos los seis intentos para adivinar la clave secreta.



La presentación de los diversos intentos, se lleva a cabo mediante cubos de siete colores diferentes.

Para la presentación de los cubos en la pantalla, se emplean los nueve gráficos definidos de la figura.



EL PROGRAMA

Los colores que maneja el Spectrum para componer la secuencia policromada van desde el negro hasta el amarillo, ambos inclusive. El blanco está descartado, debido a que el fondo de pantalla es precisamente de este color, y la presentación del mismo sería invisible. Echando cuentas vemos que los colores a utilizar son siete: negro, azul, rojo, magenta, verde, cian y amarillo.

Realmente, nuestro Spectrum no trabaja con colores propiamente dichos, sino con los códigos

¡!

Al introducir el programa, no olvidemos sustituir los caracteres subrayados, por los gráficos definidos correspondientes a las teclas indicadas.

*

Durante la ejecución del programa, si pulsamos la tecla SPACE, aparecerá en la pantalla cada uno de los intentos introducidos con anterioridad.

*

Una vez introducido el programa correctamente, podemos grabarlo mediante el comando **SAVE "CUB" LINE 1070.**

*

En las pistas facilitadas por el ordenador, un cuadrado azul representa un acierto parcial, y dos cuadraditos intermitentes en diagonal, un acierto completo.

que corresponden a los mismos, aunque esto es un detalle del programa que no nos afecta en el modo de operar con él.

Cada una de las secuencias de colores que nosotros introduzcamos, será correspondida por el Spectrum con otra secuencia de pistas:

a) Un cuadrado azul por cada color existente pero mal colocado.

b) Dos cuadritos intermitentes unidos por dos de sus vértices por cada color bien posicionado (carácter gráfico del 6).

El encuadramiento de cualquiera de las dos indicaciones anteriores es aleatorio, por lo que no debemos fiarnos del lugar que ocupen, sino del número de ellas.

En el momento en que la pantalla está saturada de información, el ordenador efectúa un desplazamiento de la misma hacia arriba *(scroll)* de cinco líneas. Esto hará que vayamos perdiendo de vista las primeras pistas.

Si durante el juego queremos saber cuáles han sido las jugadas anteriores, que han podido desaparecer de la pantalla por el efecto de *scroll* antes citado, pulsaremos SPACE (barra espaciadora en el PLUS). Inmediatamente aparecerán en la pantalla, por orden de introducción, cada uno de los intentos fallidos con su pista correspondiente.

CUB puede «recordar» un total de 40 intentos, número más que suficiente para averiguar la secuencia secreta. Esto es debido a las instrucciones **DIM I(40,4)** de la línea 60 y **DIM R(40,4)** de la línea 70, que controlan, respectivamente, las series de intentos y pistas memorizadas. Este programa puede ser utilizado tanto en el modelo de 48 K como en el de 16 K; y en ambos casos, debido a la limitación de las matrices de memorización, el programa se detendrá con el mensaje *Subscript wrong* en caso de que se llegue a rebasar este número de tope de intentos; y es que 40 intentos para adivinar cuatro colores... ¡ya está bien!

Para la introducción de los colores en nuestra combinación, utilizaremos las teclas numéricas, según el código de color que cada una de ellas tiene asignado. Hemos de tener cuidado en esta operación, ya que una vez introducido un código, no se contempla la posibilidad de su borrado. Un dato anecdótico a subrayar en el programa, es la forma en que el mensaje de autoría del programador aparece en la pantalla, así como la forma de utilizar la sentencia **RANDOMIZE** en la línea 1290 para que los cubos que aparecen en la presentación del programa sean siempre del mismo color, regenerándolos cada vez que se ejecuta el programa mediante la función **RND**, sin ocupar un espacio de memoria innecesario con una **DATA**.

Y hablando de subrayar en el programa, no olvidemos sustituir los caracteres que aparecen de este modo en el listado, por los gráficos definidos de las teclas correspondientes. En cuanto a los números doblemente subrayados, recordemos que son los gráficos cambiados de las teclas afectadas, es decir, los que se obtienen pulsando simultáneamente la tecla **CAPS SHIFT** y el número, al encontrarnos en el modo gráfico.

Para almacenar el programa en cinta, utilizaremos el comando **SAVE "CUB" LINE 1070**, con el fin de que al cargarse se produzca una autoejecución a partir de la línea 1070.

```
  10 REM ************************
  20 REM * J.M.MAYORAL SERRANO *
  30 REM ************************
  40 BORDER 7: PAPER 7: CLS
  50 REM  SPECTRUM ELIGE NUMERO
  60 LET M$='    ': LET I$='    ': DIM I(40,4): DIM M
(4): DIM D(7)
  70 DIM R(40,4)
  80 LET FILA=1
  90 LET F1=0
 100 LET SW=0
 110 FOR N=1 TO 4
 120 LET NUM=INT (RND*7)
 130 IF D(NUM+1) THEN  GO TO 120
 140 LET D(NUM+1)=1: LET M(N)=NUM: LET M$(N)=STR$ NUM
 150 NEXT N
 160 CLS
 170 REM    JUEGO
 180 FOR I=1 TO 100
 190 PRINT  PAPER 2; FLASH 1;AT 21,0;'       INTRODUC
E NUMERO'
 200 FOR C=1 TO 4
 210 LET K$=INKEY$
 220 IF INKEY$=' ' THEN  GO SUB 790
 230 IF CODE K$<48 OR CODE K$>54 THEN  GO TO 210
 240 BEEP .2,35
 250 LET I$(C)=K$: LET I(I,C)=VAL K$
 260 PRINT AT FILA,C+17;K$
 270 NEXT C
 280 LET F1=F1+1
 290 PRINT  PAPER 7;AT 21,0;'
'
 300 GO SUB 630
 310 GO SUB 420
 320 IF FILA>21 THEN  GO TO 350
 330 IF SW=1 THEN  RETURN
 340 NEXT I
 350 LET FILA=FILA-3
 360 POKE 23692,5
 370 FOR Q=1 TO 5
 380 PRINT
```

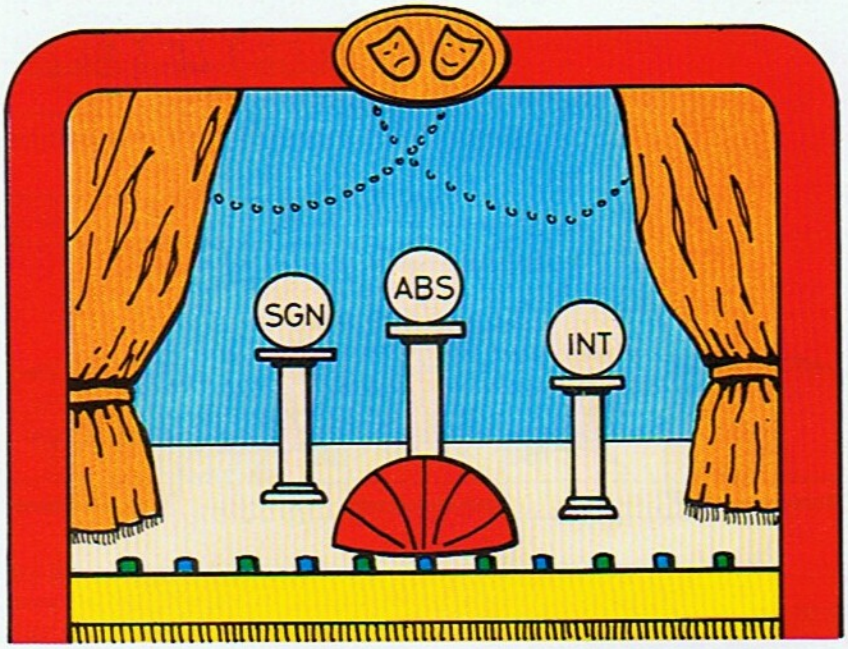
```
390 NEXT Q
400 IF SW=1 THEN  RETURN
410 NEXT I
420 REM VERIFICACION DE CODIGO
430 FOR Q=1 TO 4
440 PRINT  PAPER 7;AT FILA-3,Q+17;' '
450 NEXT Q
460 IF I$=M$ THEN  GO TO 1710
470 FOR Q=1 TO 4
480 FOR W=1 TO 4
490 IF I$(Q)=M$(W) AND Q=W THEN  LET R(F1,Q)=1
500 IF I$(Q)=M$(W) AND Q<>W THEN  LET R(F1,Q)=-1
510 NEXT W
520 NEXT Q
530 LET PL=1
540 FOR Q=4 TO 1 STEP -1
550 IF R(F1,Q)>0 THEN  GO TO 590
560 IF R(F1,Q)<0 THEN  GO TO 610
570 NEXT Q
580 RETURN
590 PRINT AT FILA-3,13+PL*2; FLASH 1; INK 2; PAPER 6
;'6'
600 LET PAS=F1*RND: LET PL=PL+1: GO TO 570
610 PRINT AT FILA-3,13+PL*2; INK 5;'8'
620 LET PL=PL+1: GO TO 570
630 REM DISPLAY JUGADA
640 LET COLUM=1
650 FOR N=1 TO 4
660 PRINT AT FILA-1,COLUM-1; INK I(F1,N);'A'
670 PRINT AT FILA-1,COLUM; INK I(F1,N);'B'
680 PRINT AT FILA-1,COLUM+1; INK I(F1,N);'C'
690 PRINT AT FILA,COLUM-1; INK I(F1,N);'D'
700 PRINT AT FILA,COLUM; PAPER I(F1,N);I(F1,N)
710 PRINT AT FILA,COLUM+1; INK I(F1,N);'E'
720 PRINT AT FILA+1,COLUM-1; INK I(F1,N);'F'
730 PRINT AT FILA+1,COLUM; INK I(F1,N);'G'
740 PRINT AT FILA+1,COLUM+1; INK I(F1,N);'H'
750 LET COLUM=COLUM+3
760 NEXT N
770 LET FILA=FILA+3
780 RETURN
790 REM   LISTADO DE JUGADAS
800 CLS
810 LET FILA=1
820 LET SW=1
830 LET FF1=F1
840 LET F1=0
850 FOR S=1 TO 4
860 LET F1=F1+1
870 GO SUB 320
880 GO SUB 630
890 GO SUB 530
900 IF F1=FF1 THEN  GO TO 930
910 NEXT S
920 GO TO 850
930 LET SW=0: LET F1=FF1
940 IF INKEY$='' THEN  GO TO 940
950 CLS : LET FILA=1
960 PRINT  PAPER 2; FLASH 1;AT 21,0;'        INTRODUC
E NUMERO
970 RETURN
980 REM  SBR. PRESENTACION
990 DATA 0,0,0,0,1,0,3,3
1000 DATA 0,0,127,255,255,0,255,255
1010 DATA 0,0,248,244,236,28,220,220
1020 DATA 3,3,3,3,3,3,3,3
1030 DATA 220,220,220,220,220,220,220,216
1040 DATA 3,3,0,0,0,0,0,0
1050 DATA 255,255,0,0,0,0,0,0
1060 DATA 208,192,0,0,0,0,0,0
1070 LET A$='ABCDEFGH'
1080 FOR N=1 TO LEN A$
1090 FOR F=0 TO 7
1100 READ A
1110 POKE USR A$(N)+F,A
1120 NEXT F
1130 NEXT N
1140 BORDER 7: PAPER 7: INK 9:
1150 CLS : GO TO 1280
1160 REM
1170 LET CO=INT (RND*7)
1180 PRINT  INK CO;AT F-1,C-1;'A'
1190 PRINT  INK CO;AT F-1,C;'B'
1200 PRINT  INK CO;AT F-1,C+1;'C'
1210 PRINT  INK CO;AT F,C-1;'D'
1220 PRINT  PAPER CO;AT F,C;' '
1230 PRINT  INK CO;AT F,C+1;'E'
1240 PRINT  INK CO;AT F+1,C-1;'F'
1250 PRINT  INK CO;AT F+1,C;'G'
1260 PRINT  INK CO;AT F+1,C+1;'H'
1270 BEEP .1,30: RETURN
1280 REM
1290 RANDOMIZE 2: DIM P(13,29)
1300 DATA 1,1,1,1,0,1,1,1,1
1310 DATA 1,0,0,1,0,1,1,0,1
1320 DATA 1,0,0,1,0,1,1,1,0
1330 DATA 1,0,0,1,0,1,1,0,1
1340 DATA 1,1,1,1,1,1,1,1,1
1350 DATA 127,32,74,46,77,46,32,77,65,89,79,82,65,76
1360 DATA 32,83,69,82,82,65,78,79
1370 FOR F=1 TO 13 STEP 3
1380 FOR C=1 TO 7 STEP 3
1390 READ A
1400 LET P(F,C)=A
1410 NEXT C
1420 FOR U=12 TO 18 STEP 3
1430 READ A
1440 LET P(F,U)=A
1450 NEXT U
1460 FOR B=23 TO 29 STEP 3
1470 READ A
1480 LET P(F,B)=A
1490 NEXT B
1500 NEXT F
1510 REM
1520 FOR F=1 TO 13 STEP 3
1530 FOR C=1 TO 29
1540 IF P(F,C)=0 THEN  GO TO 1560
1550 GO SUB 1160
1560 NEXT C
1570 NEXT F
1580 REM
1590 FOR C=1 TO 30
1600 PRINT  BRIGHT 1; INK 6; PAPER 2;AT 20,C;'3'
1610 PRINT  BRIGHT 1; INK 5; PAPER 4;AT 21,C;'3'
1620 NEXT C
1630 REM
1640 FOR N=1 TO 22
1650 READ A
1660 PRINT AT 17,N+4;CHR$ A
1670 NEXT N
1680 FOR N=1 TO 500
1690 NEXT N
1700 RUN
1710 REM  CODIGO ACERTADO
1720 GO SUB 470
1730 FOR N=1 TO 50
1740 NEXT N
1750 CLS
1760 LET FILA=10
1770 LET COLUM=10
1780 GO SUB 650
1790 PLOT 60,70
1800 DRAW 120,0
1810 DRAW 0,45
1820 DRAW -120,0
1830 DRAW 0,-45
1840 LET K=1
1850 LET FILA=7
1860 LET PL=-1: GO SUB 540
1870 PRINT  RAPER 6;AT 17,1;'LO HAS RESUELTO EN ';F1;
' INTENTOS!'
1880 PRINT  INK 4;AT 21,0;'  PULSA UNA TECLA PARA EMP
EZAR '
1890 FOR C=1 TO 31
1900 PRINT  PAPER 6; INK 9; OVER 1;AT 17,C;' '; INK 9
; PAPER K; OVER 1;AT 17,C-1;' '
1910 BEEP .01,C
1920 IF INKEY$='' THEN  GO TO 1940
1930 GO TO 1970
1940 NEXT C: PRINT  PAPER K;AT 17,31;' '
1950 LET K=K+1: IF K>5 THEN  LET K=1
1960 GO TO 1890
1970 RUN 50
```

TRATAMIENTO NUMERICO

HASTA el momento hemos hablado de algunas de las funciones de que dispone el vocabulario BASIC, concretamente, de las aplicables al tratamiento de expresiones y variables del tipo cadena *(string)*; tales son: **LEN, STR$, VAL** y **VAL$**. Sin embargo, existe otro tipo de funciones de notable importancia, aplicables de manera general a expresiones numéricas, a las que denominamos FUNCIONES NUMERICAS, cuyo funcionamiento y misión trataremos en este capítulo. El formato general de estas funciones es similar al de las ya descritas de cadena; así pues, constan de nombre de FUNCION y ARGUMENTO (constante, variable o expresión) que será evaluado para obtener el resultado.

Podemos decir por tanto, que la sintaxis general de cualquier función BASIC, sea del tipo que sea, es: FUNCION + ARGUMENTO; donde el argumento puede ser una constante, una variable o una expresión, siempre y cuando se ajuste al tipo de dato sobre el que opera la función: de cadena o numérico.

Entre las funciones numéricas, de las que nos ocuparemos en las próximas páginas dedicadas al lenguaje BASIC, se cuentan: **SGN, ABS, INT, SQR, LN, EXP** y las denominadas funciones trigonométricas (**SIN, COS, TAN, ASN, ACS** y **ATN**). Además de todas estas funciones, en su mayor parte matemáticas, hablaremos de algunos otros elementos BASIC relacionados con ellas. Pero lo mejor que podemos hacer es comenzar sin más preámbulos.



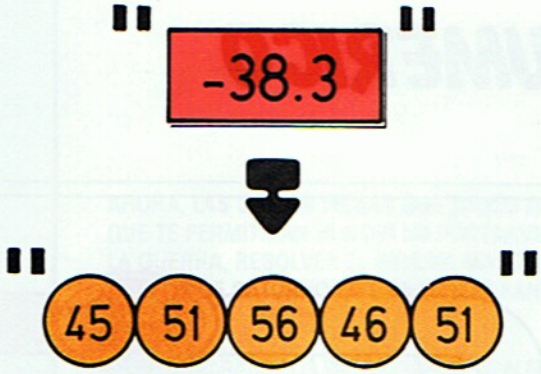
Las funciones BASIC juegan un papel muy importante en el teatro de la programación.

SGN

Por medio de esta función puede evaluarse el signo de un argumento numérico. Evidentemente, **SGN** no es aplicable a variables de cadena, y sí en cambio a constantes, variables o expresiones numéricas.

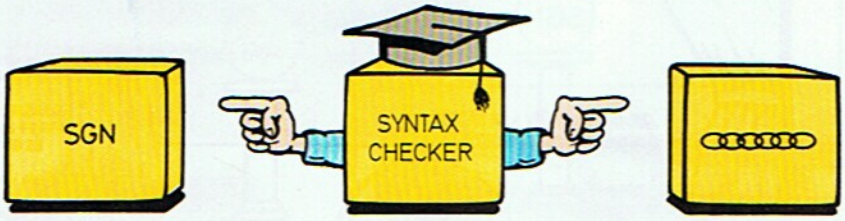
Tengamos en cuenta, que aunque dentro de una variable de cadena algunos de sus caracteres, o más concretamente el primero de ellos, puede ser un signo más (+) o menos (–), a efectos del ordenador, por encontrarse entrecomillados, éstos carecen de su función habitual de signo matemático, y no dejan de ser meros caracteres, como lo son una letra A o cualquier símbolo o signo de puntuación, tales como la admiración (!) o el dó-

SGN es la función BASIC cuya misión es descubrir el signo de su argumento.



Cada número dentro de una cadena no es más que su código A.S.C.I.I.

el syntax checker no admite que la función SGN *se aplique directamente sobre un valor de cadena.*



lar ($). Para ser más exactos, los signos más (+) o menos (–), dentro de una cadena, no son otra cosa que los caracteres cuyo código A.S.C.I.I. corresponde a 43 y 45, respectivamente.

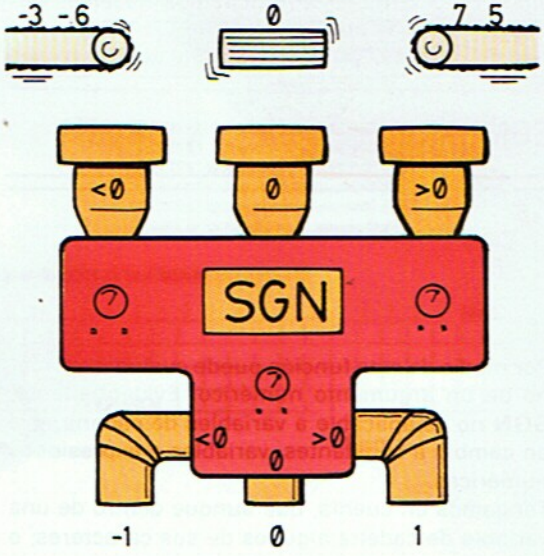
En este momento, conviene recordar la existencia de la función **VAL**, que ya vimos anteriormente, cuya misión es convertir una variable de cadena en numérica. Gracias a ella, podemos someter la cadena a un proceso previo de conversión, tras el cual poderle aplicar, ya en forma de valor numérico, la función **SGN**.

Supongamos que queremos saber cual es el signo de la constante menos tres (–3). En este caso, por tratarse de un valor numérico, podemos aplicar directamente la función **SGN** de la siguiente manera: **PRINT SGN (-3)**. Si por el contrario, deseáramos conocer el signo del literal menos tres ("-3"), deberíamos recurrir previamente a la función **VAL: PRINT SGN VAL "-3"**. Cualquier intento por evaluar el signo de una cadena, sin pasar previamente por la función **VAL**, será rechazado sin miramientos por el estricto *syntax checker* (comprobador de sintaxis). Probemos a hacerlo y observar que sucede: **PRINT SGN "-3"**. El ejemplo no ha querido servir más que como botón de muestra, de la necesidad de realizar la conversión de cadena a valor numérico para operar con la función **SGN**. Sin embargo, puede que no hallemos mucha utilidad en aplicar una función sobre una constante: es evidente que no nos hace falta un ordenador para conocer el signo de menos tres (–3); conviene, por tanto, recordar el hecho de que las constantes no tienen por qué ser tan simples como la que se ha empleado en el ejemplo. Para reafirmar esta teoría, transcribimos una constante cuya complejidad habla por sí sola: **(((-3*5+ SQR (15*3/ 2)- (28 * 5 / 3)) *2-3)-66)**. Esta vez no estaría de más el concurso del ordenador para encontrar su signo ¿verdad?

En estos casos en los que no entran en juego variables que un programa puede alterar, es conveniente que en vez de introducir en el programa el cálculo de la función, lo llevemos a cabo nosotros mismos, mediante el modo directo, y lo insertemos ya calculado en el programa. Gracias a ello, ahorraremos memoria y tiempo de ejecución, puesto que la expresión completa siempre ocuparía más espacio que su resultado, y además tendría que ser recalculada cada vez que el ordenador pasara por ese punto del programa; operación bastante inútil si tenemos en cuenta que al tratarse de una expresión constante, su valor nunca variará.

Como hemos dicho al comienzo de este epígrafe, la función **SGN** indica el signo de su argumento, pero ¿de qué manera lo señala? Quizá podríamos pensar que simplemente mediante un carácter menos ("–") o más ("+"). Sin embargo, esta solución no es válida: en primer lugar hemos de notar que el nombre de la función es **SGN** (abreviatura de la palabra inglesa *SiGN,* signo), por tanto debe devolver un valor numérico, puesto que de no ser así, como ya hemos visto al tratar las funciones de cadena, su denominación debería terminar en un símbolo dólar ($), siendo por tanto **SGN$**.

Los valores devueltos por la función SGN *son: –1 para argumentos negativos, 1 para argumentos positivos y 0 para argumento cero.*



¡!

La función **SGN** devuelve el signo de su argumento, codificando la información según el siguiente convenio: –1 = argumento negativo, 0 = argumento cero, 1 = argumento positivo.

*

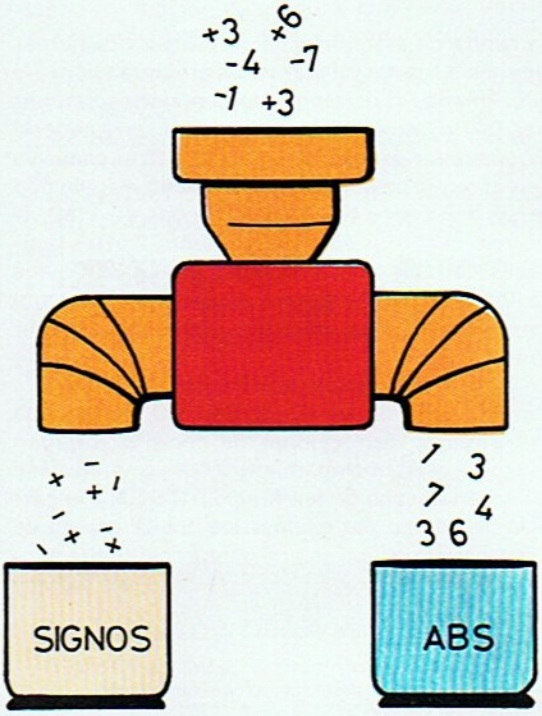
En el orden de prioridades, todas las funciones tienen la misma entre sí, por lo que se efectúan de izquierda a derecha.

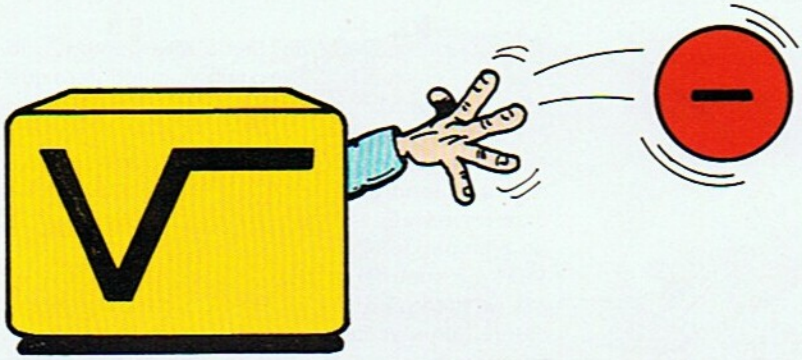
Por otra parte, el ordenador siempre proporciona una mayor facilidad de manejo con datos numéricos, por lo que suelen ser más fácilmente utilizables los datos de este tipo que los de cadena. Así pues, ¿cuál es el convenio que se adopta para indicar un signo mediante números?: como resultado de la función obtenemos tres valores posibles; 1 si el argumento es positivo, 0 si es cero y -1 si el argumento es negativo. Veamos algunos ejemplos:

```
10 REM - SGN
20 LET A=-2:LET B=0:LET C=3
30 PRINT SGN A,SGN B,SGN C,SGN
   A/B,SGN A/C,SGN C/A
```

La utilidad de esta función puede que todavía no se nos manifieste claramente, pero según vayamos avanzando en la programación, iremos tomando conciencia del servicio que presta. No olvidemos que algunas operaciones matemáticas, incluso de uso muy común, como la radicación, no admiten valores negativos. Por ejemplo, si tenemos una instrucción del tipo **10 PRINT SQR A**, corremos el riesgo de que la variable A adopte un valor negativo; de ser así, al ejecutarse la línea 10 el programa se detendría con un error del tipo **A Invalid argument**. Una forma de evitar esto, sería utilizar la función **SGN**, para no dejar llegar a la operación nada más que valores positivos o cero: **10 IF SGN A>=0 THEN PRINT SQR A.**

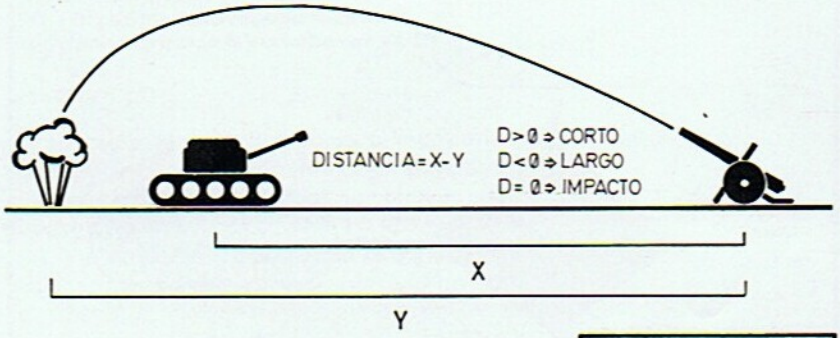
La función **ABS** *priva a los números de su signo.*





La radicación, al igual que algunas otras operaciones matemáticas, no admite un operando negativo.

Planteamiento de la rutina "TIRO AL TANQUE". La variable D contiene la distancia que separa el tanque de la posición de impacto del proyectil.



En el caso concreto del ejemplo, hubiera sido más sencillo emplear directamente la comparación sobre A (**10 IF A>=0 THEN PRINT SQR A**), aunque sin duda, mediante **SGN** el programa gana en claridad, en cuanto a las intenciones del programador sobre esa línea del programa. En algunas otras ocasiones no es sólo conveniente o clarificador, sino prácticamente inevitable recurrir a **SGN**.

ABS

Esta función nos permite obtener el valor absoluto de un número. Matemáticamente, entendemos por obtener el valor absoluto privar de su signo al número afectado; así pues, el valor absoluto de tres (|3|; el valor absoluto de un número se representa convencionalmente entre barras verticales), es igual al de menos tres (|-3|). Para ser más concretos, la obtención del valor absoluto,

¡!

La función **ABS** calcula el valor absoluto de su argumento; para ello, convierte en positivo su argumento, cualquiera que sea la magnitud o signo original de este.

*

Un sistema sencillo para averiguar si determinado número es o no entero, consiste en comparar su forma original y su forma entera, ya sea mediante el operador "igual que" (=), "distinto que" (<>) o simplemente restando sus valores numéricos (-).

más que desproveer del signo al argumento, lo que hace es forzarlo a ser positivo, cualquiera que fuera su signo anterior.

De lo dicho anteriormente, se desprende que al aplicar la función **SGN** sobre un valor absoluto, nunca obtendremos el valor –1. La utilidad de esta función es también muy grande; pongamos un ejemplo de ello:

Vamos a suponer que en el diseño de un programa de juego, en el que se intenta alcanzar mediante un proyectil un tanque en movimiento, deseamos indicar al usuario lo alejado que quedó el último disparo de su objetivo, con el fin de corregir la trayectoria en la próxima andanada. Llegando a este punto veremos que se pueden producir dos circunstancias: o bien el disparo quedó largo o bien corto (desecharemos las posibilidades de izquierda y derecha para no complicar el ejemplo).

En la variable D, tendremos el valor de la posición del tanque menos la de impacto del proyectil. Si suponemos que el origen de coordenadas se encuentra en nuestra base, cuando la variable D resulte positiva, el disparo habrá sido corto; de ser negativo, largo, y siendo cero, un éxito completo. Pasemos ahora a la transcripción efectiva del planteamiento a un programa BASIC:

```
10 REM D CONTIENE LA DISTANCIA AL
   BLANCO
20 IF D=0 THEN PRINT "** FELICIDADES
   **"'"IMPACTO DIRECTO": GO TO 70
30 PRINT "EL DISPARO HA QUEDADO ";
40 IF SGN D>0 THEN PRINT "CORTO"
50 IF SGN D<0 THEN PRINT "LARGO"
60 PRINT "POR ";ABS D;" METROS."
70 REM CONTINUACION DEL PRO-
   GRAMA
```

Con esta sencilla rutina, se imprimirán en la pantalla los mensajes indicando el resultado del disparo. Indudablemente, habremos mejorado la presentación de datos del programa, imprimiendo, por ejemplo:

```
EL DISPARO HA QUEDADO LARGO
POR 15 METROS
```

que no habiendo presentado D en su valor con signo (sin utilizar **ABS**), como sería el caso de:

```
EL DISPARO HA QUEDADO A -15 METROS
```

La rutina de ejemplo, nos da pie para comentar algunas características de la programación, no relacionadas concretamente con ciertas sentencias BASIC, sino más bien con el diseño general de cualquier programa. Por una parte, como ya hemos visto, podíamos haber evitado la inclusión de las funciones **SGN** en las líneas 40 y 50, si bien es cierto que sólo ahorraríamos 2 bytes (unidades de memoria), que entre los 16.384 bytes de los que como mínimo dispone el Spectrum para nuestro uso, resultan bastante insignificantes.

De mayor importancia es la forma de disponer las sentencias **IF** de las líneas 40 y 50, puesto que este tipo de instrucciones de decisión retardan bastante el funcionamiento de un programa. Como en el caso de las funciones **SGN** , la pérdida de tiempo por evaluar los dos **IF** carece de importancia, si bien es una buena costumbre ahorrar memoria y tiempo de ejecución, en todos y

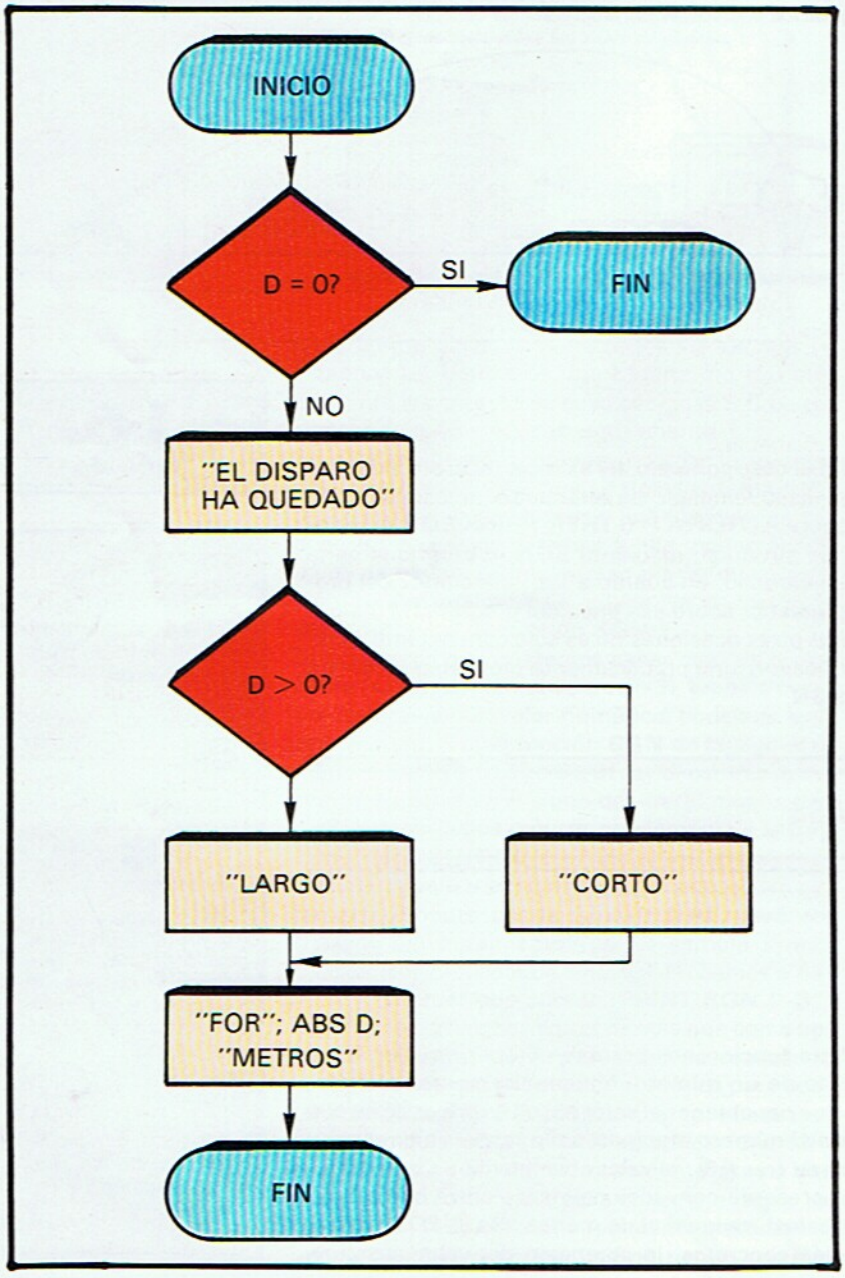


Diagrama de flujo del programa "TIRO AL TANQUE", con la estructura de decisiones exclusivas evitada mediante un puente.

cada uno de los elementos de la programación. En la totalidad de un programa de relativa longitud, la diferencia de ese conjunto de pequeños ahorros, sí se hace sentir.

En el caso concreto de la rutina que nos ocupa, encontramos en dos líneas de programa seguidas (40 y 50), lo que en programación BASIC se conoce generalmente como **IF** exclusivos o DECISIONES EXCLUSIVAS. Este término implica que dadas dos decisiones consecutivas, cuando una de ellas se cumple, la otra no puede cumplirse bajo ninguna circunstancia.

En nuestro caso, dado que el posible valor cero de la variable D ha sido tratado ya en la línea 20, al llegar a las instrucciones 40 y 50 del programa, nos encontramos con dos decisiones exclusivas: **D>0** y **D<0**. Evidentemente, si se cumple que D es mayor que cero, jamás se puede cumplir en la línea siguiente (puesto que no ha habido alteración del valor de D), que D sea menor que cero, y viceversa.

Para la eliminación de este tipo de estructuras, nos apoyamos en la sentencia GO TO, construyendo lo que denominamos un "puente". En el caso concreto de nuestro ejemplo, "puentearemos" la línea 50 de programa, mediante un **GO TO** a la 60 al finalizar la primera comparación, y eliminaremos el **IF** ya innecesario en la línea siguiente.

De esta forma, sólo se realizará una comparación (la de la línea 40), ya que si ésta se cumple, se imprimirá el mensaje correspondiente y pasaremos directamente a la línea 60; por el contrario, si el **IF** de la línea 40 no se hubiera cumplido, se pasaría directamente a la instrucción 50, donde se tiene libre acceso a la impresión del mensaje sin la evaluación de ninguna condición. Veamos el programa bajo su nueva configuración:

```
10 REM D CONTIENE LA DISTANCIA AL
   BLANCO
20 IF D=0 THEN PRINT "** FELICIDADES
   **""""IMPACTO DIRECTO": GO TO 70
30 PRINT "EL DISPARO HA QUEDADO ";
40 IF D>0 THEN PRINT "CORTO": GO TO
60
50 PRINT "LARGO"
60 PRINT "POR ";ABS D;" METROS."
70 REM CONTINUACION DEL PROGRA-
MA
```

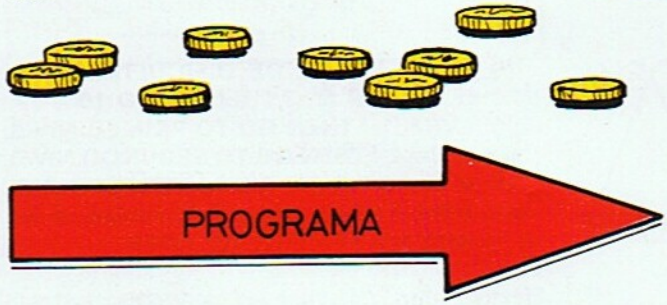
Mediante una técnica de "puente" basada en la sentencia **GO TO**, *podemos ahorrar espacio de memoria y tiempo de ejecución, evitando la evaluación de un* **IF**.

Gracias a esta sencilla técnica, hemos conseguido reducir la evaluación innecesaria de dos condiciones a una sola. Ello nos ha obligado a incluir una nueva instrucción: **GO TO 60**, en la línea 40. Sin embargo, ha merecido la pena la modificación, puesto que hemos ahorrado en espacio de memoria, y lo que es más importante, en tiempo de ejecución.

Es de destacar la considerable diferencia en cuanto al tiempo que el ordenador emplea en tratar una sentencia **IF**, respecto a una **GO TO**. Nos bastará saber que el espacio que en la "inteligencia" del Spectrum (su memoria ROM) se utiliza para el tratamiento de los **IF**, es decenas de veces superior al empleado para la ejecución de un **GO TO**.

Depende de nosotros que, según las condiciones particulares de cada programa, decidamos ahorrar memoria (suprimiendo por ejemplo sentencias **SGN** que puedan ser obviadas), o aumentar

Los pequeños ahorros de memoria y tiempo de ejecución en un programa, conducen al final del mismo a un ahorro total considerable.

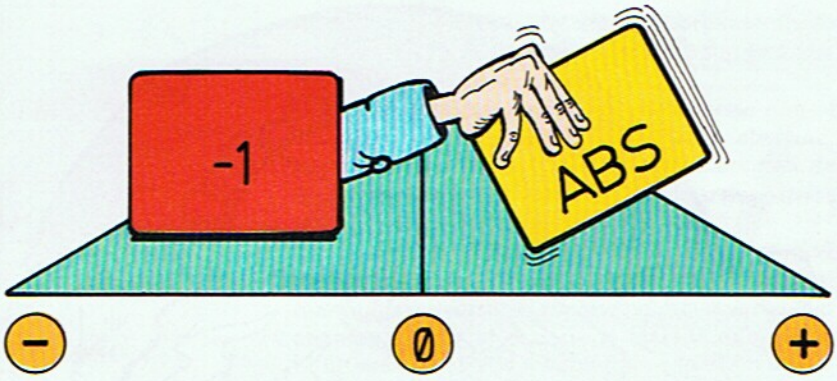


¡!

Combinando la función **ABS** con la multiplicación por el elemento neutro negativo, podemos cerciorarnos de que un determinado valor no es positivo. Para ello, hallaremos el valor absoluto del número en cuestión, y multiplicaremos este por –1.

*

El sistema de "puenteado", para evitar estructuras de decisiones exclusivas, debe ser utilizado siempre que aparezca un problema de este tipo.



La operación producto con el elemento neutro, por el valor absoluto de un número, siempre da como resultado el número original con signo negativo.

la claridad del mismo. No obstante, la técnica del "puente" en los **IF** exclusivos es recomendable en todo momento.

Otra utilidad de la función **ABS** puede ser la de asegurarse la obtención de un número negativo; esto se puede conseguir forzando previamente el signo del número a positivo (mediante **ABS**), y multiplicando posteriormente este resultado por **-1: PRINT -1*ABS N.** Al multiplicar por -1, no alteramos la magnitud (valor numérico sin signo) del número, puesto que 1 es el elemento neutro de la operación producto; además de lo cual, cambiamos el signo de la expresión al multiplicar por un número negativo. Si esta operación hubiera sido realizada sin efectuar anteriormenete el valor absoluto, correríamos el riesgo de encontrar un número originalmente negativo y cambiarlo a signo positivo.

Finalizaremos estas líneas de estudio de la función **ABS** (del inglés *ABSolute*, absoluto), haciendo notar que su uso sólo afecta al signo del argumento, y en ningún momento a la magnitud del mismo. Probemos con algunos ejemplos:

```
10 REM - ABS
20 LET A=-2:LET B=0:LET C=3
30 PRINT ABS A,ABS B,ABS C,ABS
   A/B,ABS A/C,ABS C/A
```

INT

Entre las funciones del BASIC del Spectrum dedicadas al tratamiento numérico, que no constituyen una operación matemática, probablemente sea **INT** la de uso más frecuente. La misión de esta función es encontrar el valor entero de su argumento, ya sea este entero o decimal, positivo o negativo, despreciando sus decimales.

Aunque actualmente los microordenadores consideran como habitual el uso de valores numéricos con coma flotante (números decimales), gran cantidad de sucesos representables mediante números y que se pueden tratar en una computadora, son valores enteros. Por ejemplo, podríamos citar los números de teléfono, los valores que se pueden obtener al lanzar un dado, etc...

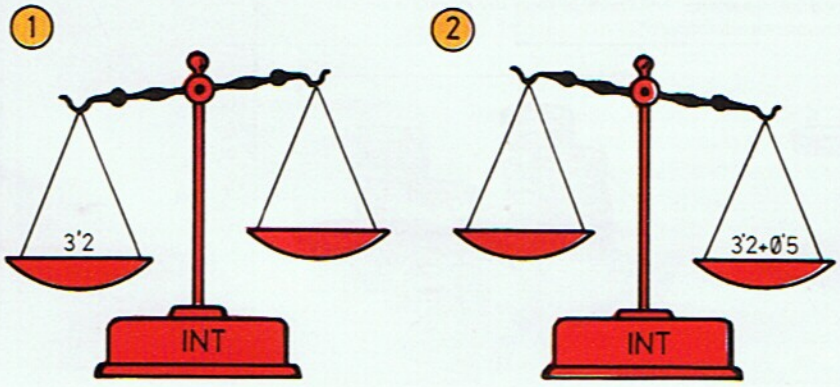
Debido a esto, es muy interesante disponer de una forma de cerciorarnos que el número sometido a tratamiento es entero y no decimal. Como el resto de las funciones, **INT** no modifica su argumento, sino que simplemente obtiene su parte entera. Del mismo modo, tampoco es de importancia el hecho de que el argumento de la función sea ya entero, puesto que aunque aparentemente no se produzca efecto alguno, la tarea de conversión a entero será llevada a cabo, y no se producirá ningún tipo de detención por error.

Generalmente, **INT** se utiliza durante las tomas de datos, para adaptar al formato entero determinadas entradas. Supongamos, por ejemplo, que durante la realización de un juego, interrogamos al usuario sobre el nivel de dificultad que desea escoger. Una depuración evidente es la del rango máximo que puede tener este dato, por ejemplo, entre 1 y 3. Observemos el siguiente programa:

```
10 INPUT "NIVEL DE DIFICULTAD ? ";N
20 IF N<1 OR N>3 THEN GO TO 10
30 IF N="1" THEN GO TO PRIMER NIVEL
40 IF N="2" THEN GO TO SEGUNDO NIVEL
50 IF N="3" THEN GO TO TERCER NIVEL
60 REM PARTE DEL PROGRAMA A LA
   QUE NO SE DEBE ACCEDER
```

Efectivamente, en la línea 20 se ha realizado una depuración para que no entren valores superio-

El "peso" de 0.5 hace que la función INT *se decante por un redondeo comercial.*



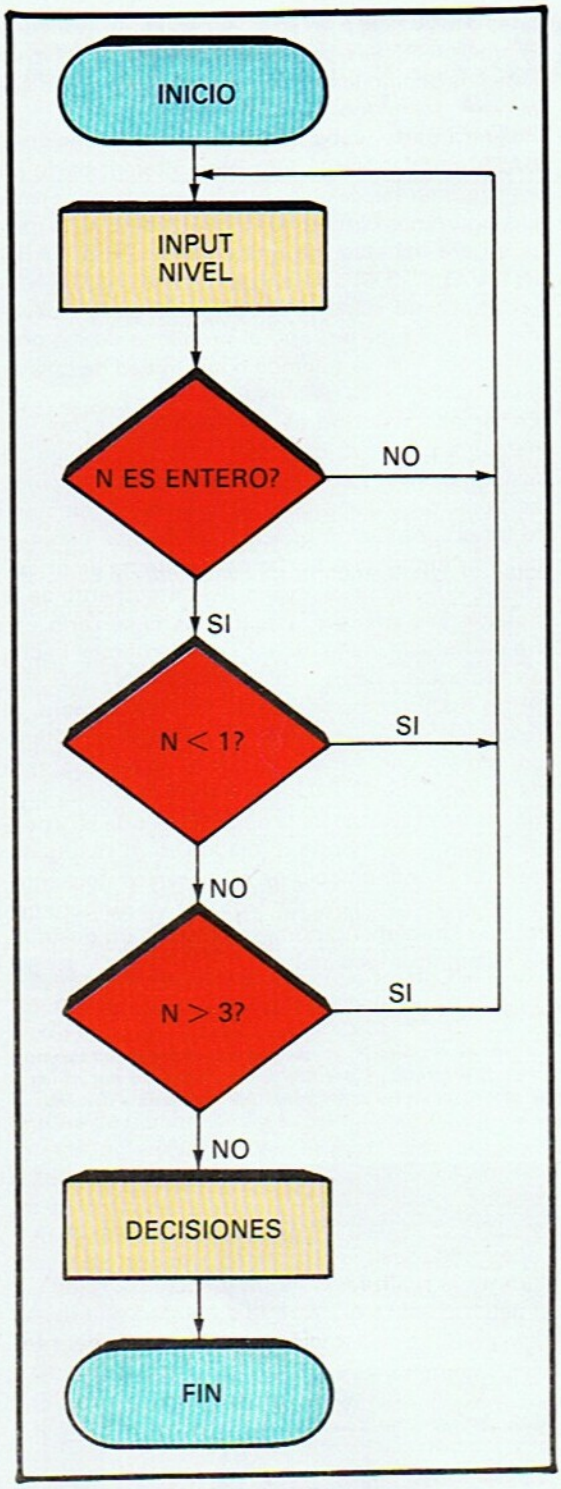


Diagrama de flujo de la rutina de depuración de entrada, una vez que le ha sido aplicado el control de números decimales.

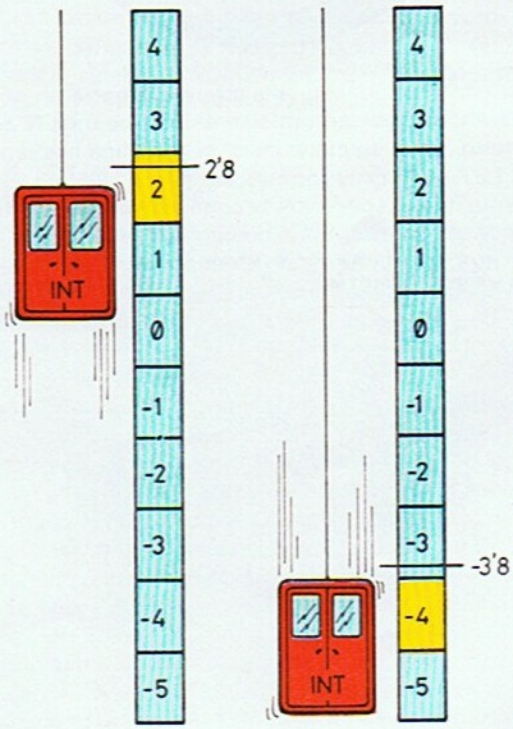
res a tres ni inferiores a uno, ya que al no cumplir ninguna de las condiciones del bloque 30-50, pasarían a la zona reservada de programa, a partir de la 60.

Sin embargo, qué sucederá si el usuario introduce un valor decimal, ya sea por descuido o por poner a prueba nuestra habilidad como programadores. Si este valor supera a tres, o es inferior a uno, será capturado por nuestra depuración de la línea 20, pero ¿y si no es así?; supongamos que el número introducido es 2.3. El valor erróneo pasará la instrucción 20, al estar comprendido entre uno y tres, y además, no se bifurcará en ninguna de las condiciones del bloque 30-50, puesto que no es ni uno, ni dos, ni tres. Evidentemente, el programa ejecutará la instrucción 60, con el consiguiente error.

Gracias a la función **INT**, nos aseguraremos de que el valor que llega a la línea 20 sea entero, de forma que no evite la depuración. Para ello, podemos seguir dos sistemas: o bien realizar una depuración en la línea 15 de valores enteros, o bien modificar el valor de N, para convertirlo incondicionalmente en entero. Las líneas podrían quedar como sigue: **15 IF N<>INT N THEN GO TO 10** o **15 LET N=INT N.**

En el primer caso, como depuración de si el valor es entero o no, se emplea la condición SI N ES DISTINTO DEL VALOR ENTERO DE N, puesto que sólo N será igual a su valor entero, cuando efectivamente sea entero. Otro sistema habría sido codificar la instrucción como: **15 IF N-INT N THEN GO TO 10** (equivalente a **15 IF N-INT N<>0 THEN GO TO 10**), el algoritmo se basa

La función INT *efectúa un redondeo por defecto, es decir, obtiene el entero inferior más próximo a su argumento.*



¡!

La función **INT** toma la parte entera de su argumento, redondeando siempre por defecto, es decir, al entero inferior más próximo.

*

Las funciones BASIC no alteran sus argumentos, sino que simplemente obtienen determinado valor en base a ellos.

*

En base a la función **INT** podemos realizar operaciones de redondeo comercial, es decir, al entero superior aquellos valores que superen o igualen a un medio sobre su parte entera, y al inferior los restantes; para realizar esta operación, sumaremos 0.5 al valor a redondear, y posteriormente hallaremos la parte entera del resultado obtenido.

La función INT *corta la parte decimal de su argumento, extrayendo su parte entera.*

¡!

Las funciones BASIC pueden servir a su vez como argumentos de otras funciones, bien sea individualmente, o como parte de expresiones más complejas.

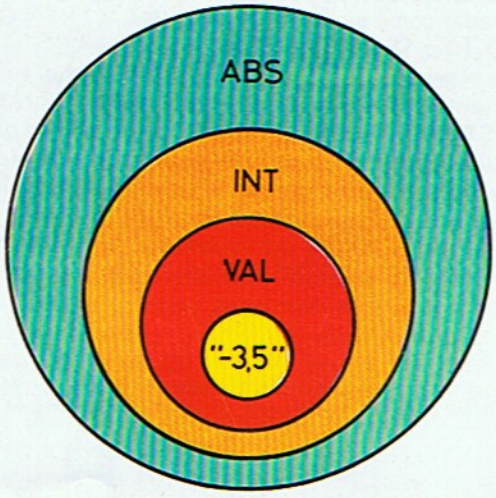
*

El ahorro de memoria y tiempo de ejecución, por pequeño que parezca, conduce, al cabo de un programa de relativa longitud, a un efecto de ahorro sensible.

*

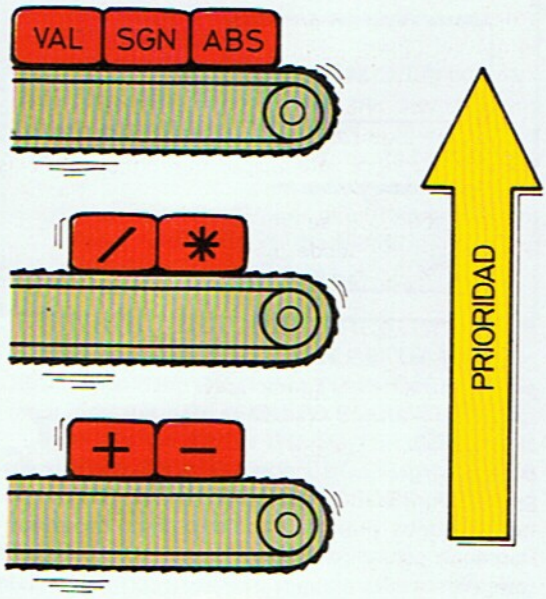
En base a la función **INT**, podemos construir una estructura que diferencia los números pares de los impares: si la división del número entre dos da un resultado entero, el número será par; de no ser así, será impar (**IF NUMERO/2=INT (NUMERO/2) THEN PRINT "PAR"**).

Todas las funciones BASIC, pueden utilizar como argumento otras funciones, siempre y cuando sean del tipo adecuado (numéricas o de cadena).



en esta ocasión en que la diferencia entre un número y su parte entera sólo es cero cuando **N** es entero. En el segundo caso, se modifica mediante **LET** el valor **N** introducido por el usuario, de forma que aseguramos la condición de entero de este valor que llega a la depuración de la línea 20.
En cuanto a la forma de obtenerse la parte entera, hemos de tener muy presente que la función **INT** redondea por defecto, es decir, toma el valor del número entero menor más próximo. Así, por ejemplo, el valor entero que **INT** obtiene del argumento 2.3 es 2, mientras que el obtenido de -2.3 es -3.
Cuando en el tratamiento numérico para aplicaciones comerciales, deseamos obtener no la parte entera sino el redondeo de un número, la función **INT** sólo nos serviría directamente en caso de cantidades con un decimal inferior a 5, puesto que cantidades decimales por encima de éste serán también redondeadas al número inferior, en vez de al superior, como es nuestra intención. Para subsanar este problema, el sistema es sumar 0.5 al argumento de **INT**. Así por ejemplo, la línea **PRINT INT (N+.5)**, escribirá el redondeo "comercial" de N, cualquiera que sea el valor de éste (los decimales mayores o iguales a 0.5 al superior, y los menores al inferior).
Por otra parte, cabe destacar que las funciones BASIC pueden ser argumentos o tomar parte en los argumentos de otras funciones. Así, por ejemplo, podremos obtener el valor absoluto de la parte entera del valor de una cadena: **PRINT ABS INT VAL "-3.5"**. Al entrar en este tipo de expresiones de relativa complejidad, pasa nuevamente a primer término el problema de las preferencias, que ya tuvimos oportunidad de aplicar a las operaciones matemáticas.
En el caso concreto de las funciones, estas tienen una prioridad muy alta, sólo por debajo de los paréntesis, y el troceado de cadenas *(slicing)*, así como de la asignación de subíndices (elemento de programación que aún no hemos estudiado). Por tanto, para burlar este sistema de prioridades, cuando deseemos que el argumento de la función sea múltiple, deberemos encerrarlo entre paréntesis, de la manera que lo hemos hecho en la función de redondeo (**PRINT INT (N+.5)**). Si no hubieramos procedido de esta manera, el resultado no se parecería en absoluto al obtenido, puesto que, en vez de hallar la parte entera de la suma de N y 0.5, se habría realizado previamente el cálculo de la parte entera de N, y posteriormente se habría sumado 0.5 al resultado; de esta forma, no sólo no hubieramos conseguido un redondeo, sino tan siquiera un número entero (la suma de un número entero y un decimal, es siempre un decimal).

Las funciones BASIC se encuentran dentro de un mismo nivel de prioridad, y por tanto, son evaluadas por orden de aparición en las expresiones (de izquierda a derecha).



EL MAPA DEL SPECTRUM

EN el capítulo anterior hicimos un buen repaso de los principales componentes de nuestro Spectrum, aclarando diversos aspectos sobre sus características, cometido, etc... Así pues, ha llegado la hora de poner una nota gráfica sobre el tema, finalizando esta etapa de nuestro conocimiento del ordenador con un completo mapa de la situación general de los principales componentes.

Nos consta que entre los aficionados a la microinformática, se encuentra un gran número de valiosos «manitas», que ya han hecho algún que otro pinito en el mundillo del *hardware;* especialmente dedicadas a ellos van las próximas páginas, en las que hacemos un rápido recorrido por el interior de nuestro ordenador.

Por tanto, aquellos que no seamos expertos en el manejo de destornilladores, soldadores, martillos... (no, no... martillos, no), más vale que nos abstengamos de poner en práctica las sugerencias de las páginas siguientes (convenientemente señalizadas con un PROHIBIDO INEXPERTOS y FIN DE PROHIBICION), y pasemos directamente al mapa de la doble página que finaliza el artículo, la cual conseguirá que los «excursionistas» del *hardware* no «perdamos el norte», si en alguna ocasión vemos un Spectrum tal cual, ¡como Sir Clive Sinclair lo trajo al mundo!

Antes de comenzar este *tour* hemos de hacer algunas aclaraciones: en primer lugar, conviene saber que nuestro ordenador está protegido por la garantía del Distribuidor Oficial: cualquier manipulación en el interior del aparato nos hace perder todo derecho a una reparación gratuita; por tanto, antes de romper el sello de garantía, tengamos muy en cuenta las posibles consecuencias de esta acción.

En segundo lugar, se dan breves indicaciones sobre los puntos desde los que podemos fabricar una salida de monitor para nuestro Spectrum; aunque esta operación se puede considerar muy sencilla, y consiste simplemente en la soldadura de dos cables a las entradas del modulador de U.H.F., no es recomendable, si no nos sentimos suficientemente capaces, aventurarnos con un soldador entre la nube de componentes de nuestro Spectrum.

El hablar de las memorias, nos da pie para aclarar una pequeña confusión que se puede producir sobre las características del Spectrum 48 K y el PLUS: la única diferencia existente entre ambos es el teclado, y un pequeño pulsador de *RESET* que se conecta a la placa mediante dos cablecillos. Excepción hecha de estas dos distinciones, y por supuesto las inherentes a pertenecer a diferentes series de producción (forma y situación de la placa disipadora de calor, distribución de pistas en la placa, etc.) ambos modelos son exactamente iguales.

Por último añadiremos que los Spectrum en su interior no son todos exactamente iguales, puesto que a lo largo del proceso de fabricación se han ido estableciendo determinadas mejoras que han dado lugar a diferentes versiones, aunque sólo desde el punto de vista *hardware,* puesto que externamente y a efectos de programación son iguales (salvo los valores devueltos por la función **IN**). Estas diferencias consisten en el trazado de las pistas, situación y forma de la placa disipadora de calor, etc. o incluso la posibilidad de que los modelos de 16 K sean ampliados internamente a 48 K, ya que en la primera versión del aparato, puesta a la venta por correo en el Reino Unido, esta posibilidad no existía, no pudiendo ser ampliables los modelos de 16 K.

Si nos pica la curiosidad de conocer la versión de nuestro Spectrum, encontraremos su nombre grabado en el anverso de la placa, en su zona inferior derecha (a la izquierda del altavoz), escrito tras la palabra *ISSUE* (serie).

He aquí los protagonistas de nuestra aventura: el Spectrum y el Plus.

¡!

Si no tenemos experiencia en electrónica, lo mejor que podemos hacer es dejar el interior del Spectrum para los profesionales.

*

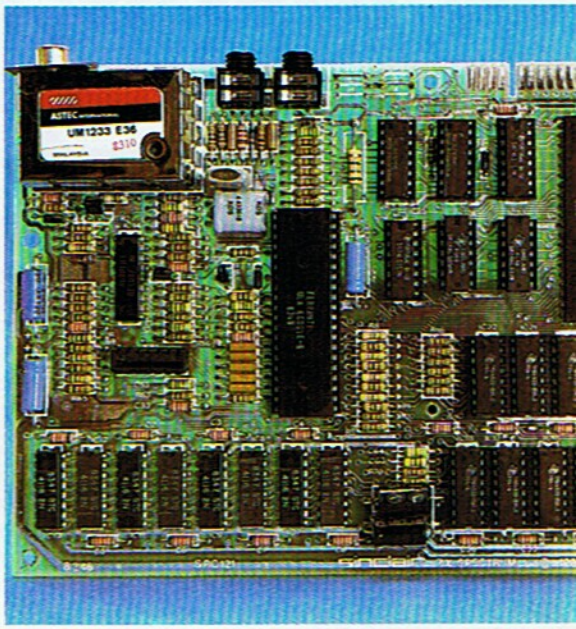
A aquellos que no nos atrevamos a poner nuestro ordenador en la «mesa de operaciones», nos bastará echar un vistazo al mapa general que finaliza esta sección; en él encontraremos una fácil referencia sobre la situación aproximada de los principales componentes del Spectrum.

*

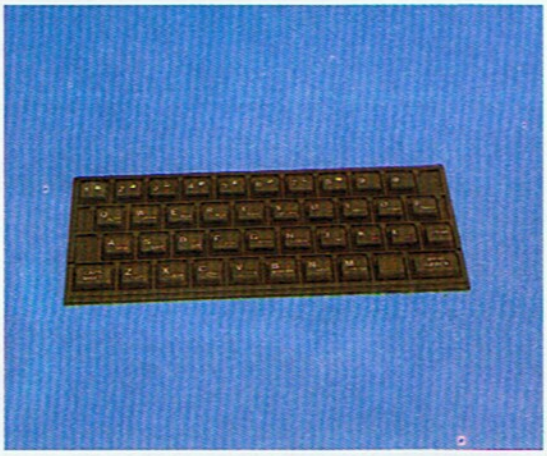
Recordemos que al romper el sello de garantía que se encuentra sobre el tornillo superior central, perdemos cualquier derecho a la reparación gratuita del ordenador.

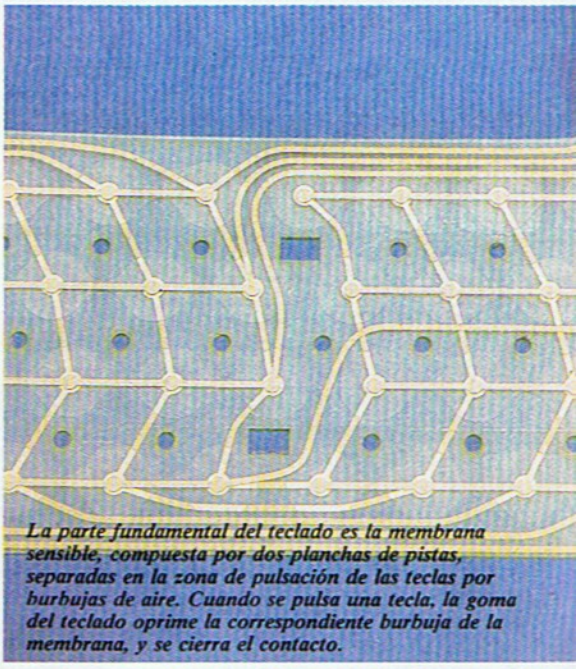
INEXPERTOS

Si nuestro Spectrum no ha sido abierto nunca, el sello de garantía cerrará el paso hasta el tornillo superior central, sin quitar el cual no es posible acceder al interior del aparato.

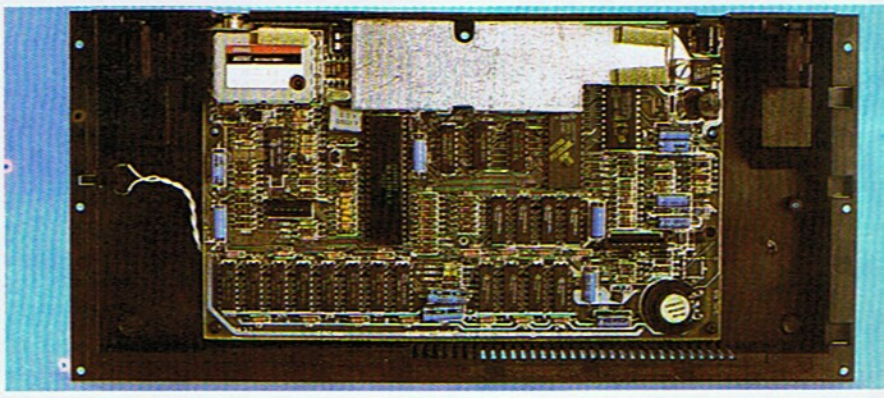


La parte del teclado en contacto con el usuario es una plancha de goma; como podemos apreciar en la foto, se trata de un elemento continuo, en el cual las teclas son bloques sobresalientes de la superficie. Por tanto, al pulsar una tecla, esta no se desliza, sino que se «hunde» junto con las inmediaciones de la plancha.





La parte fundamental del teclado es la membrana sensible, compuesta por dos planchas de pistas, separadas en la zona de pulsación de las teclas por burbujas de aire. Cuando se pulsa una tecla, la goma del teclado oprime la correspondiente burbuja de la membrana, y se cierra el contacto.

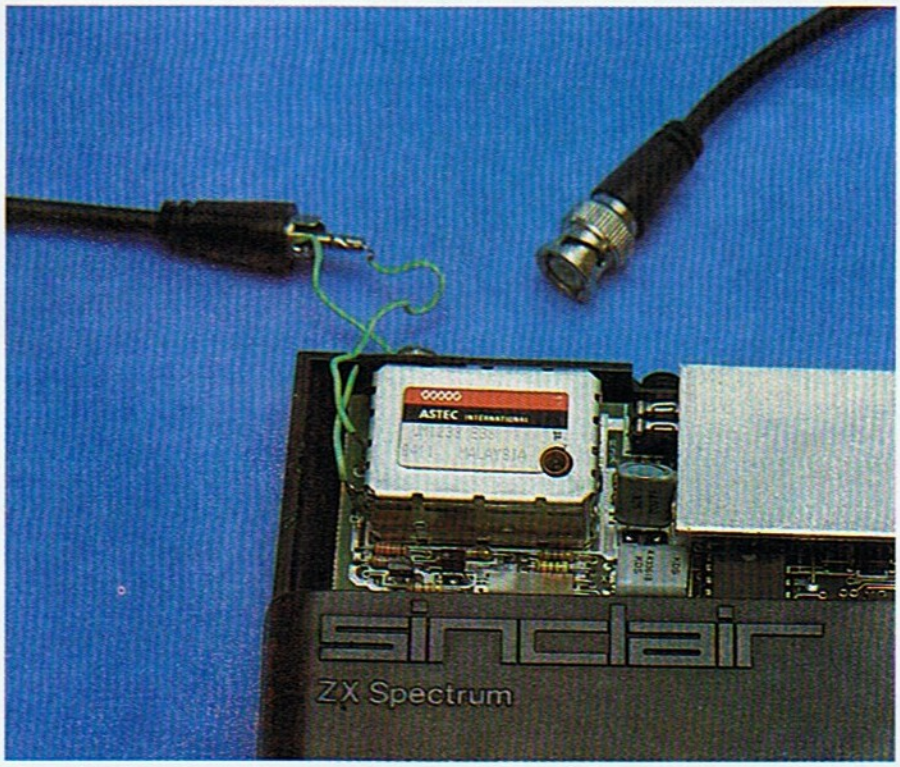


Un par de cablecillos de color blanco, unidos a un pulsador en el lateral izquierdo de la caja, constituyen el sistema de RESET del Spectrum Plus. En la parte inferior derecha de la placa (a la izquierda del altavoz), aparece serigrafiado el tipo de versión de que disponemos.

Al Spectrum que se observa en la foto, se le ha desprovisto de su placa disipadora de calor, con el fin de poder apreciar claramente todos y cada uno de los componentes del aparato.

Una vez desmontado el Spectrum, se divide en dos partes: la superior, con todo lo relacionado con el teclado; y la inferior, que contiene la placa. La unión entre ambas la efectúan los dos cables planos que parten del teclado, y se ajustan a presión en los conectores correspondientes en la placa. En la parte superior se aprecian cuatro elementos superpuestos consecutivamente: carcasa plástica, membrana sensible, teclado y placa metálica.

obablemente una de operaciones más ncillas a realizar en el erior de un Spectrum, fabricar una salida de onitor. Sólo es cesario soldar dos bles a los conductores entrada al modulador, cuales serán soldados u vez al conector del ble del monitor.



Mapa aproximado del interior del Spectrum.

1. Modulador de televisión.
2. Altavoz
3. Fuente de alimentación
4. Memoria RAM
5. Conectores del teclado
6. Reloj
7. *Port* de comunicación exterior
8. Chips lógicos
9. Interface para casete
10. U.L.A.
11. C.P.U.
12. Memoria ROM

BIBLIOTECA
ENTRADA
SALIDA
9
7
3
11
12
5
2

ANTIAEREO

EL radiotelescopio de Arecivo ha detectado cientos de extraños objetos en dirección a la Tierra. Se cree que dichos artefactos voladores proceden de alguna estrella del cúmulo globular de Hércules. Vienen en formación de combate y dispuestos para la lucha.

Un radioaficionado ha captado una conversación entre dos naves. La posterior decodificación de dicha comunicación ha sacado a la luz las verdaderas intenciones de los extraterrestres: quieren llevarse a su planeta todos los Spectrum de la tierra, para su posterior destrucción, privando así a la juventud terrícola de uno de sus pasatiempos preferidos.

Los ejércitos de todo el mundo se han movilizado para hacerles frente. En lugares estratégicos de la corteza terrestre se han situado barreras defensivas, con lanzaderas de proyectiles de fisión nuclear, para tratar de evitar un Spectrumcidio en masa.

Nuestro deber es destruir la totalidad del convoy extraterrestre enemigo, y preservar la vida electrónica de nuestro gran amigo. Todo esto es lógicamente una ficción, ¿pero qué tal si nos vamos entrenando por si llega el momento...?

EL PROGRAMA

El objetivo del juego es abatir el mayor número de OVNIS posible, con los 50 proyectiles de que disponemos. Hay que tener en cuenta que, como

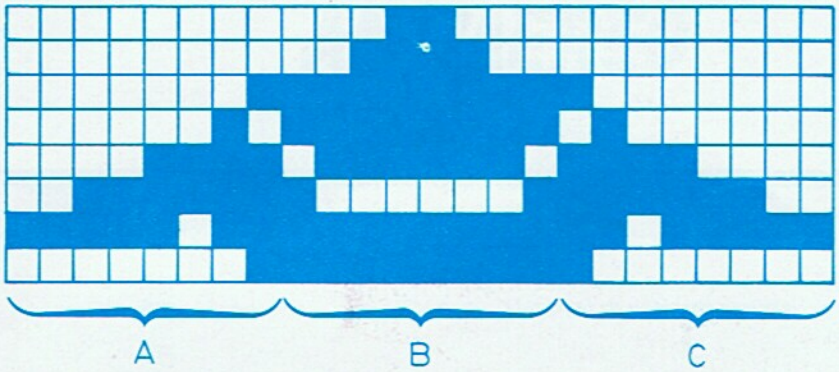
Dependiendo de la dificultad del blanco, el programa nos otorga más o menos puntos, a la hora de hacer impacto sobre un Spectrumcida extraterrestre.

1 2 3

es habitual, hasta que los misiles no alcancen el tope superior de la pantalla, no se puede volver a disparar, por lo que es importante que conservemos unos nervios de acero para hacer frente al bélico compromiso.
La puntuación que se obtiene por cada impacto está en relación a la dificultad de hacer blanco; así, por ejemplo, tendrá más valor destruir un OVNI cuanto más arriba de la pantalla aparezca; análogamente, proporciona mayor puntuación utilizar los proyectiles de los bunkers 1 y 2, que los del 3 y, a su vez, los del uno más que los del 2. Para el lanzamiento de proyectiles, emplearemos las teclas 1, 2 y 3, atendiendo a la lanzadera desde la que deseemos disparar.

Para el proyectil de fisión nuclear, se ha definido especialmente el gráfico de la letra D.

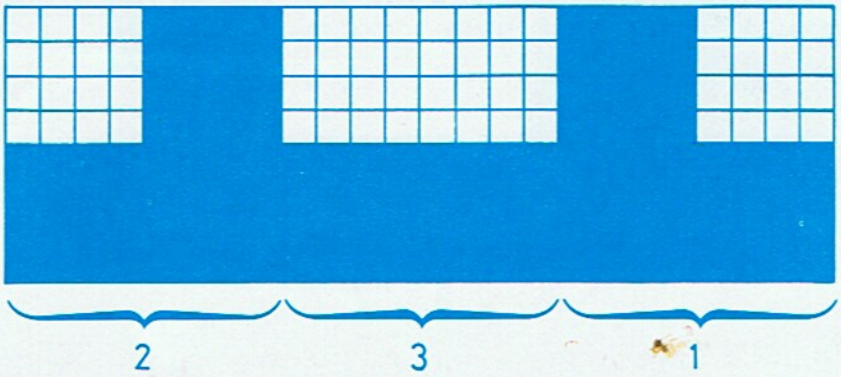
Al finalizar el juego, el Spectrum compara el número de puntos obtenidos con el record que se tenga establecido hasta el momento, informándonos sobre la pericia demostrada en relación con la de nuestros predecesores.
Este juego es sin duda uno de los clásicos de la microinformática, y por su sencillez, sirve en muchas ocasiones de ejemplo para mostrar la estructura de un programa de acción. Es una buena idea que aquellos que queráis empezar a programar, utilicéis un programa con este esquema, al cual podréis ir añadiendo, según el ánimo con que conteis, todas las mejoras que consideréis oportunas.
A la hora de la introducción del programa, conviene recordar que las letras que aparecen subrayadas en el listado, corresponden a los graficos definidos de las teclas subrayadas. Si además el subrayado fuera doble (hecho que sólo encontraremos en los números), el gráfico es el que se obtiene pulsando simultáneamente **CAPS SHIFT**, lo que se conoce como GRAFICO CAMBIADO.
Veamos un ejemplo práctico: al llegar a la línea 338, encontraremos una **D** subrayada; debere-



La nave extraterrestre está compuesta por tres gráficos definidos, correspondientes a las teclas A, B y C.

mos por tanto pasar el ordenador a modo **GRAPHICS (CAPS SHIFT + 9)**, y a continuación pulsar la tecla **D**; esto hará aparecer en la pantalla dicha letra mayúscula, no habiendo surtido efecto aparente la introducción de **GRAPHICS**; no debemos preocuparnos por ello, puesto que una vez ejecutado el programa, el ordenador se ocupará de cambiar esa **D** por el correspondiente proyectil de fisión nuclear. Una vez que hayamos introducido el gráfico, deberemos abandonar el modo **GRAPHICS (CAPS SHIFT + 9)**, para poder continuar con la introducción normal del listado.

La grabación del programa se efectuará mediante el uso combinado de los comandos **SAVE** y **LINE** de la siguiente forma: **SAVE "INVASION" LINE 630**.



Las lanzaderas de proyectiles, se han simulado, mediante los gráficos predefinidos que corresponden a las teclas 2, 3 y 1.

```
 10 REM ***********************
 20 REM *   F.LOPEZ MARTINEZ  *
 30 REM ***********************
 40 REM * Adapt. J.M. MAYORAL *
 50 REM ***********************
 60 REM
 70 LET R=0: POKE 23658,8
 80 BORDER 0: PAPER 0: INK 9
 90 CLS
100 PRINT AT 21,0; PAPER 6;'
                           '
110 PRINT AT 20,5; INK 2;'231';TAB 15; INK 3;'231';T
AB 25; INK 4;'231'
120 PRINT  PAPER 1; BRIGHT 1;AT 0,0;'PUNTOS: 0','MUN
ICION: 50';'          '
130 PRINT  PAPER 8; OVER 1;AT 21,6;1;AT 21,16;2;AT 2
1,26;3
140 LET D=0
150 LET P=0
160 LET M=50
170 LET A=INT (RND*14)+2
180 FOR L=0 TO 28
190 PRINT AT A,L;' ABC'
200 BEEP .0015,10
210 IF D THEN  GO TO 300
220 IF INKEY$='' THEN  GO TO 400
230 IF INKEY$<'1' OR INKEY$>'3' THEN  GO TO 400
240 LET K$=INKEY$: LET K=VAL K$+1
250 LET B=10*VAL INKEY$-4
260 LET D=1
270 LET M=M-1
280 LET H=19
290 PRINT  PAPER 1; BRIGHT 1;AT 0,26;'  ';AT 0,26;M
300 LET H=H-1
310 IF H<>A THEN  GO TO 330
320 IF B>L AND B<L+4 THEN  GO TO 430
330 PRINT  INK K; BRIGHT 1;AT H,B;'D';AT H+1,B;' '
340 BEEP .0015,50
350 IF H>2 THEN  GO TO 400
360 LET D=0
370 PRINT AT H,B;' '
380 IF NOT M THEN  GO TO 530
390 GO TO 400
400 NEXT L
410 PRINT AT A,29;'   '
420 GO TO 170
430 PRINT AT H+1,B;' '
440 PRINT AT A,L;'BOOUM'; INK K; BRIGHT 1;;AT A,L-1;
'A';AT A,L+5;'C'
450 FOR Q=1 TO 10
460 BEEP .1,40
470 NEXT Q
480 LET D=0
490 PRINT AT A,L-2;'          '
500 LET P=P+(16-A)*(4-((B-6)/10+1))
510 PRINT  PAPER 1; BRIGHT 1;AT 0,8;P
520 IF M THEN  GO TO 170
530 IF P<=R THEN  GO TO 570
540 PRINT AT 9,3;'ESTABLECIO UN NUEVO RECORD'
550 LET R=P
560 GO TO 580
570 PRINT AT 9,9;'EL RECOPD SIGUE'
580 PRINT AT 11,16-(10+LEN STR$ R)/2;'EN ';R;' PUNTO
S'
590 PRINT AT 13,3;'DESEA INTENTARLO DE NUEVO ?'
600 IF INKEY$='S' THEN  GO TO 80
610 IF INKEY$<>'N' THEN  GO TO 600
620 GO TO 10000
630 REM  GRAFICOS DEFINIDOS
640 DATA 0,0,1,2,15,63,251,1
650 DATA 24,60,255,255,126,129,255,255
660 DATA 0,0,128,64,240,252,223,128
670 DATA 16,16,16,56,108,40,40,40
680 LET A$='ABCD'
690 FOR N=1 TO LEN A$
700 FOR F=0 TO 7
710 READ A
720 POKE USR A$(N)+F,A
730 NEXT F
740 NEXT N
750 RUN
```

FUNCIONES MATEMATICAS

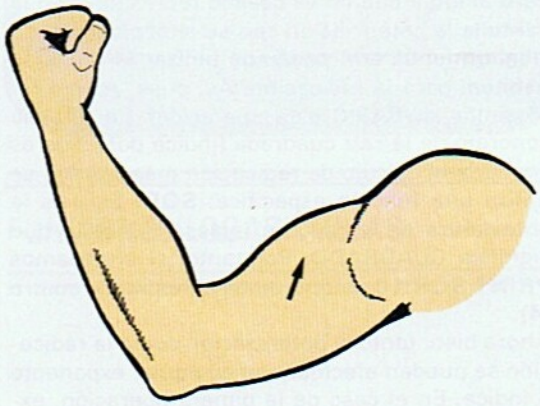
BAJO esta denominación general, trataremos las funciones de potenciación, exponenciación, logarítmicas y trigonométricas. La estructura de estas funciones es muy similar a las de tratamiento numérico que vimos anteriormente; no obstante, su uso es bastante menos frecuente, al restringirse su aplicación al campo científico o de cálculo matemático.

Por ello, no debemos asustarnos si en algún momento no llegamos a comprender exactamente las explicaciones sobre la utilización de estas funciones. Tengamos en cuenta que, si realmente no conocemos su uso, nunca las utilizaremos en nuestros propios programas. Por tanto, nos bastará con saber que existen, así como su sintaxis, para facilitarnos la comprensión e introducción de listados de programas, confeccionados por otras personas que sí las utilicen.

Así pues, los poco introducidos en el campo de las matemáticas, no deben preocuparse por no comprender completamente las explicaciones que a continuación se brindan. En todo caso, se ha procurado simplificar estas en lo posible, lo que puede dar lugar a algunas imprecisiones que no son dignas de ser tenidas en consideración.

POTENCIACION

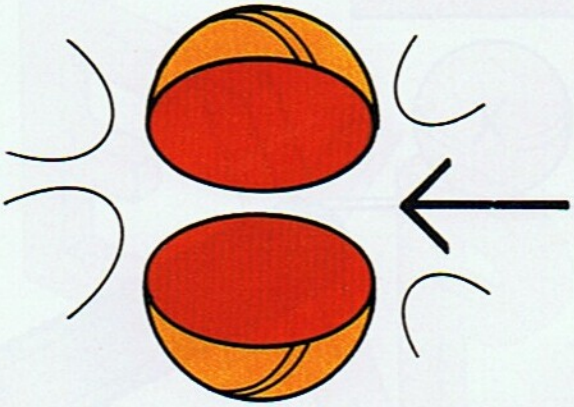
Es de todos conocido que la potenciación consiste en multiplicar un número, denominado base, por sí mismo, tantas veces como indica el superíndice que le sucede, que se denomina exponente. Así por ejemplo, el término 4^2, que se lee "cuatro elevado a dos" (o "cuatro al cuadrado"), tiene como resultado 4×4=16. Lógicamente, debido a la restricción de no poder utilizar la notación habitual de superíndices, en BASIC se adopta el criterio de representar esta operación mediante la base, seguida de un signo "flecha hacia arriba" (↑) y, a continuación, el exponente. Con lo cual, la operación del ejemplo anterior se escribiría en BASIC de la siguiente manera: 4 ↑ 2.



Para la obtención de la función de potenciación, en el Spectrum se utiliza un símbolo «flecha hacia arriba» (↑).

Dentro de las funciones matemáticas, probablemente sea esta la de uso más frecuente, sobre todo con el exponente dos. Daremos un breve repaso a nuestros conocimientos básicos sobre esta operación.

La potenciación, al igual que el resto de las operaciones matemáticas que hemos visto hasta el momento (suma, multiplicación, etc.), tiene una operación inversa la cual nos permite conocer qué base elevada a un exponente proporciona determinado resultado; esta operación se denomina radicación, y se representa por una uve (V) en cuya parte superior se expresa el índice de la raíz, unida a una línea de super-rayado debajo de la cual se escribe el radicando (argumento de la función).



La potenciación admite exponentes fraccionarios, mediante los cuales conseguir la función de radicación.

¡!

La función exponencial y logarítmica son inversas.

*

Para la obtención en el Spectrum de logaritmos en una base distinta de e, tenemos que utilizar una propiedad de los logaritmos, por la cual, el logaritmo en base B de N, es igual al cociente del logaritmo natural de N, partido por el logaritmo natural de la base B.

Los argumentos de las funciones trigonométricas deben expresarse obligatoriamente en radianes.

La potenciación consiste en multiplicar un número (base) por sí mismo, tantas veces como indica el superíndice (exponente) que le sigue.

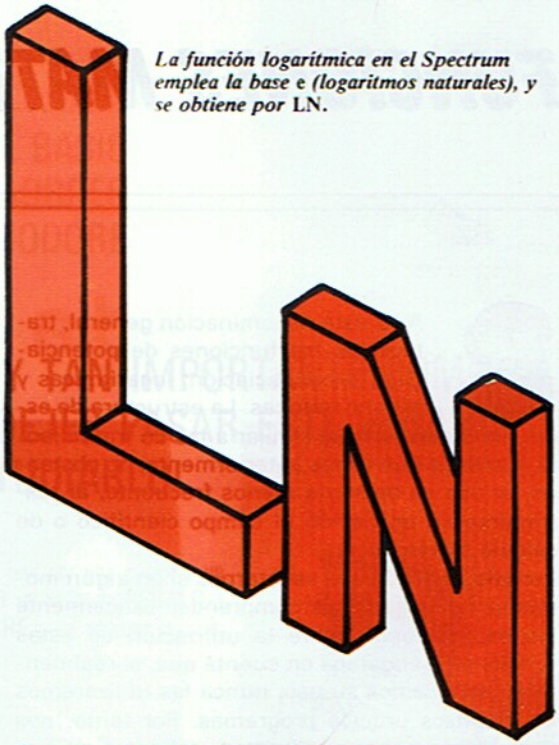
En el caso concreto del ejemplo anterior, la operación inversa al cuadrado de cuatro, es la raíz cuadrada de dieciséis, representada $\sqrt{16}$, cuyo resultado es 4. Así pues, se cumple que dada una base cualquiera **B**, y un exponente cualquiera **E**...

$$B^E = X \text{ y } B = \sqrt[E]{X}$$

Pero al igual que no es posible representar en la pantalla la potenciación con su simbología tradicional, menos aún podemos utilizar la notación habitual para la radicación. Así pues, ¿cómo representa en BASIC esta operación? En el caso concreto de la raíz cuadrada (índice dos), que es sin duda el cálculo de radicación más común, se utiliza una función específica: **SQR**. Esta es la abreviatura de la palabra inglesa *SQuaRe,* que significa CUADRADO. Por tanto, si efectuamos **PRINT SQR 16** obtendremos el resultado cuatro **(4)**.

Ahora bien, tanto la potenciación como la radicación se pueden efectuar con cualquier exponente o índice. En el caso de la pimera operación, expresar un exponente distinto a dos, es bien sencillo: no hay más que cambiar el número escrito tras la flecha por el exponente que deseemos; sin embargo, en el caso de la radicación, la función BASIC **SQR** implica una raíz cuadrada, ¿cómo expresar entonces un índice distinto de dos? Para responder a esta pregunta es preciso disponer de una cierta base matemática.

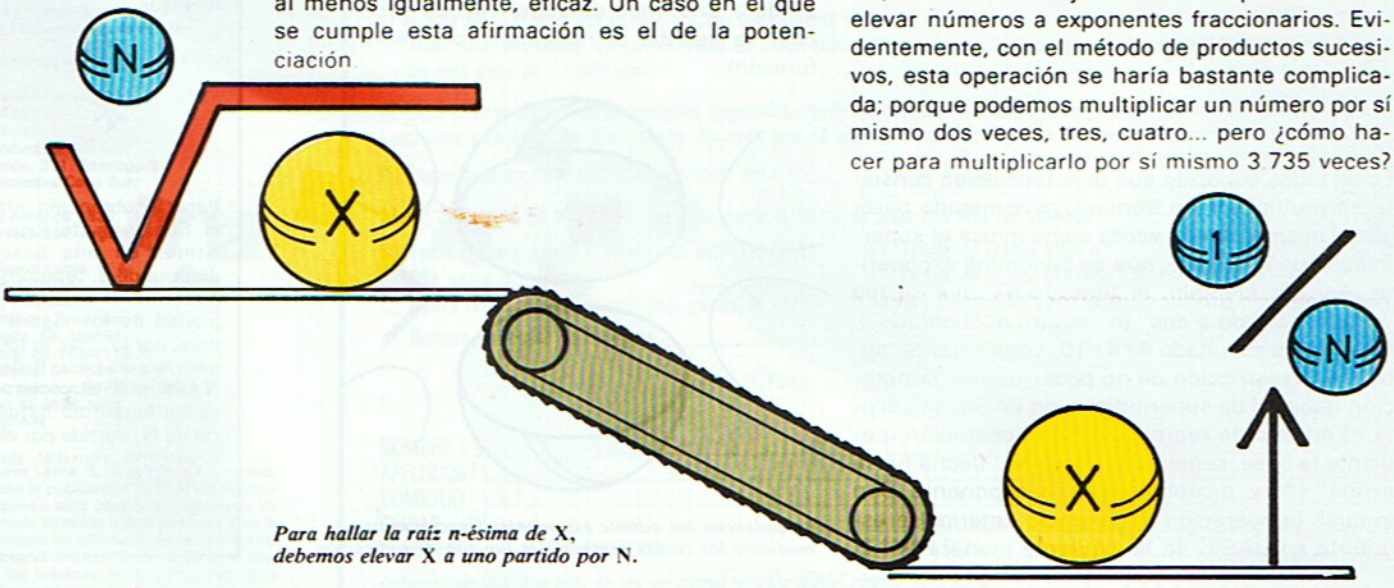
Los conocimientos de cálculo matemático de nuestro Spectrum son considerables, y prácticamente es seguro que, sin ánimo de menospreciarnos, superen los que nosotros tenemos. Por ello, la forma en que el ordenador realiza determinadas operaciones internamente, no se corresponde exactamente con la manera tradicional por la que las llevaríamos a cabo los profanos en la materia, aunque el sistema empleado sea más, o al menos igualmente, eficaz. Un caso en el que se cumple esta afirmación es el de la potenciación.



La función logarítmica en el Spectrum emplea la base e (logaritmos naturales), y se obtiene por LN.

Como ya hemos dicho, elevar una base a un exponente equivale a multiplicar dicha base por sí misma, tantas veces como indique el exponente; sin embargo, esta no es la forma en la que el Spectrum lleva a cabo dicha operación. Nuestro ordenador se basa en su capacidad para realizar la operación matemática denominada LOGARITMO, que dentro de muy poco pasaremos a estudiar. Más concretamente, el sistema de potenciación se apoya en una propiedad de los logaritmos, por la cual la base **B** elevada al exponente **E**, es equivalente al antilogaritmo del exponente **E** multiplicado por el logaritmo de la base. Esto se expresa en BASIC de la siguiente manera: **B ↑ E = EXP (E*LN B)**.

Seguir este sistema para el cálculo de la potencia, tiene una ventaja clarísima: la posibilidad de elevar números a exponentes fraccionarios. Evidentemente, con el método de productos sucesivos, esta operación se haría bastante complicada; porque podemos multiplicar un número por sí mismo dos veces, tres, cuatro... pero ¿cómo hacer para multiplicarlo por sí mismo 3.735 veces?



Para hallar la raíz n-ésima de X, debemos elevar X a uno partido por N.

Una buena regla mnemotécnica para recordar el valor de **e**, *la proporciona el hecho de que su aproximación coincida con 2.7, seguido del año de la muerte de Goya (1.828).*

La pregunta que se plantea inmediatamente, se refiere a la utilidad de elevar una base a un exponente fraccionario; pues bien, es mucha, y pronto lo comprobaremos, puesto que nos permite obtener las raíces de cualquier índice a través de la potenciación.

El hecho antes mencionado, se fundamenta en una propiedad de las potencias, por la cual una base cualquiera **B**, elevada a un exponente fraccionario **N/D**, es igual a la raíz de índice **D** de la base **B** elevada a **N**, o lo que es lo mismo:

$$B^{N/D} = \sqrt[D]{B^N}$$

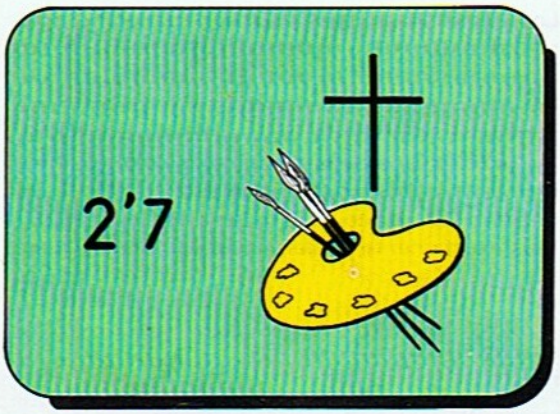
Por tanto, puesto que cualquier base elevada a exponente 1 tiene por resultado la misma base, podemos particularizar la expresión anterior, asegurando que la raíz n-esima de cualquier número es igual al número elevado a uno partido por el índice. Esto quiere decir, que la obtención de la raíz cuadrada se puede llevar a cabo de dos formas diferentes: una por medio de la función BASIC específica **(SQR)** y otra, elevando la base a un medio **(1/2)**. Gracias a este sistema, podemos obtener cualquier raíz; así por ejemplo, la raíz de índice tres de 343 se puede averiguar mediante la siguiente instrucción: **PRINT 343↑(1/3)**.

Para finalizar con el tema de las potencias, y puesto que hemos visto como obtener las radicaciones mediante esta operación, vamos a enumerar algunas propiedades interesantes, que no están directamente relacionadas con el BASIC, pero que nos pueden ser de utilidad a la hora de realizar algún programa que precise esta operación:

B↑0 = 1
B↑(-E) = (1/B)↑E
B↑(E+F) = B↑E * B↑F
B↑(E*F) = B↑E↑F

En cuanto a la prioridad en el orden de ejecución, la potenciación tiene la más alta entre las operaciones matemáticas elementales. Otra precaución a tomar para la correcta ejecución de las potenciaciones, es recordar que la utilización de bases negativas produce un error del tipo **A Invalid argument**, al igual que ocurre con la función **SQR**, como tuvimos oportunidad de comprobar en el capítulo anterior cuando estudiamos la función **SGN**.

La inversa de la función logarítmica es la función exponencial, obtenida mediante **EXP**.



FUNCIONES LOGARITMICAS

Comenzaremos este epígrafe definiendo la función que trata: el LOGARITMO en cualquier base **B** de un número **N**, es el exponente al que se ha de elevar **B** para obtener **N**. Se trata pues de una de las operaciones inversas a la potenciación; por tanto, hemos de tener cuidado de no intentar efectuar el logaritmo de un número negativo, para no obtener de nuevo el desagradable mensaje de **A Invalid argument** (argumento erróneo). Los logaritmos, como hemos visto, se pueden expresar en cualquier base, pero sin duda son dos las más ampliamente utilizadas en el cálculo matemático: la base 10 y la base **e**. Los de la base 10, son también conocidos bajo el nombre de logaritmos decimales, vulgares o de BRIGGS, por ser este matemático el que confeccionó la primera tabla de logaritmos de este tipo.

¡!

La función de potenciación (↑), permite utilizar exponentes fraccionarios, gracias a lo cual facilita la obtención de raíces enésimas.

*

Algunas de las propiedades de las potencias, que conviene recordar son: **A↑0=1**, **A↑(-B)=(1/A)↑B**, **A↑(B+C)=A↑B*↑A↑C**, **A↑(B*C)=A↑B↑C**.

*

La fórmula general para la obtención de la raíz n-ésima de **B** es: **B↑(1/N)**.

¡!

Los logaritmos obtenidos directamente por el Spectrum son los del tipo neperiano o natural (base **e**).

*

El Spectrum incorpora la constante **PI** internamente, con un valor aproximado 3.1415927.

*

La función exponencial (antilogaritmo) se obtiene en el Spectrum mediante **EXP**.

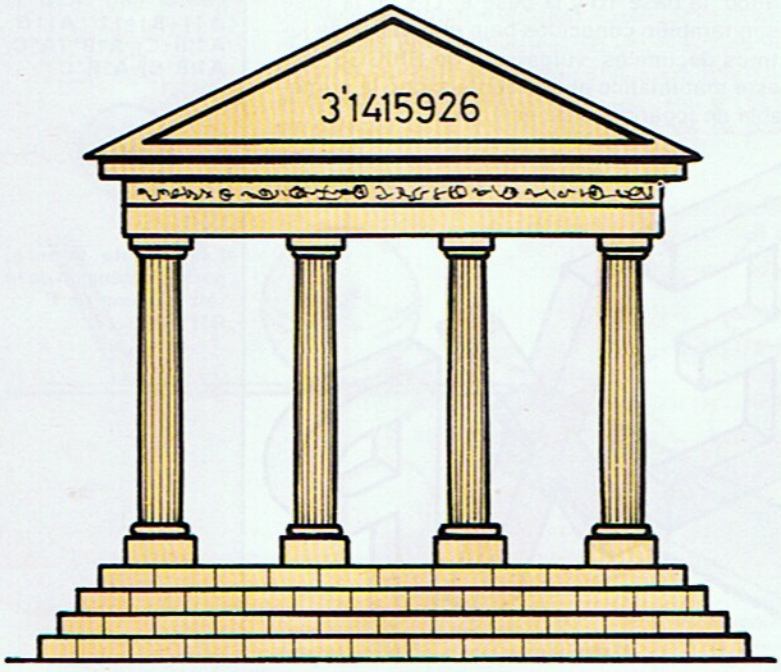
Los logaritmos en base **e**, se conocen como logaritmos naturales o neperianos, en memoria del matemático JOHN NEPER. El número **e**, de forma análoga a **PI**, es una constante matemática ampliamente utilizada, cuyo valor aproximado es 2.7182818. Una regla mnemotécnica muy difundida para la memorización de esta constante, es añadir a 2.7, el año de la muerte de Goya (1828). Más exactamente, la definición del número **e** es:

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

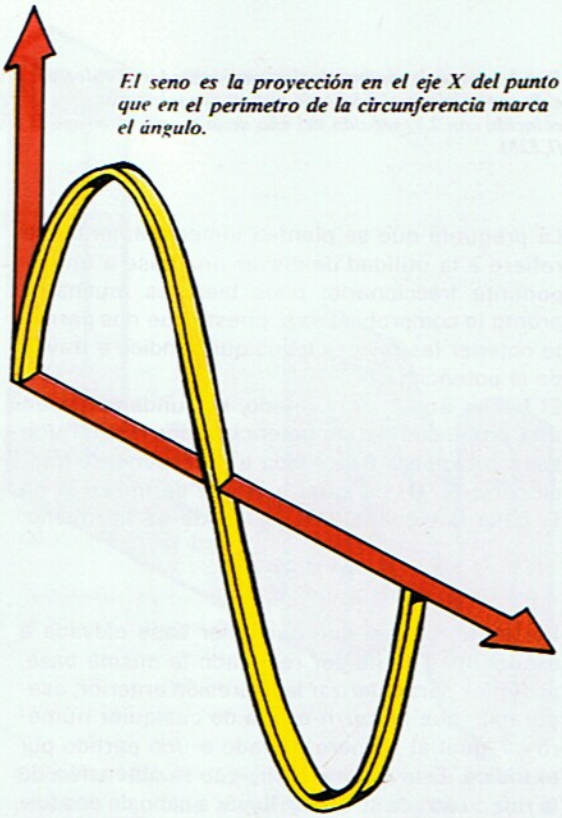
Ahora bien, el punto que más nos interesa en esta sección, es la forma en que nuestro Spectrum es capaz de llevar a cabo esta operación; en este sentido, hemos de hacer observar que el ordenador únicamente es capaz de hallar directamente logaritmos neperianos, que se obtienen mediante la función BASIC **LN**. Este tipo de logaritmos son empleados bastante frecuentemente en los cálculos matemáticos de una cierta complejidad; sin embargo, para el usuario con menos aspiraciones no reportan una gran utilidad.

Seguramente, sólo deseamos saber, a qué exponente hay que elevar 3 para que dé el resultado 243, es decir, obtener el logaritmo en base 3 de 243. No debemos preocuparnos por este pequeño detalle; podremos obtener logaritmos en cualquier base, apoyándose en la siguiente propiedad de los logaritmos:

El valor de la constante PI (π) se conoce ya desde la Antigua Grecia.



El seno es la proyección en el eje X del punto que en el perímetro de la circunferencia marca el ángulo.



Dadas dos bases **B** y **B'**, el logaritmo en base **B'** de cualquier número **(N)**, es igual al logaritmo de **N** en base **B**, partido por el logaritmo de **B'** en base **B**:

$$\text{Log}_{B'}\ N = \frac{\text{Log}_B\ N}{\text{Log}_B\ B'}$$

Según esta propiedad, conocido el logaritmo de un número en cualquier base, en nuestro caso **e**, podemos saber el logaritmo en cualquier otra, simplemente dividiendo el resultado del logaritmo en la base conocida, por el logaritmo de la base en que se desea conocer. Si aplicamos esta regla a nuestro ejemplo, tendremos que:

$$\text{Log}_3\ 243 = \frac{\text{LN } 243}{\text{LN } 3}$$ => **PRINT LN 243/LN 3**

Aún nos queda por hablar de un par de propiedades más de los logaritmos, que quizás nos sean de utilidad en la simplificación de algunos cálculos con estas operaciones: el logaritmo de una potencia es igual al exponente multiplicado por el logaritmo de la base de la potencia **(Log B ↑ E = E * Log B)**; el logaritmo de la base logarítmica es siempre la unidad **(Log_N N = 1)**; y el logaritmo de uno es cero **(Log_N 1=0)**.

Finalmente, nos resta hablar de la función **(EXP)**, que realiza la operación inversa al logaritmo (antilogaritmo). Gracias a ella, podremos obtener el número cuyo logaritmo produce determinado resultado. Utilizando esta función, podemos escri-

bir la aproximación del número **e** que el ordenador emplea en sus cálculos: según la propiedad vista anteriormente, el logaritmo de la base logarítmica es siempre la unidad, por tanto el antilogaritmo de uno debe dar siempre como resultado la base logarítmica; así pues, para obtener **e**, base de los logaritmos neperianos, sólo debemos escribir el antilogaritmo neperiano de la unidad:

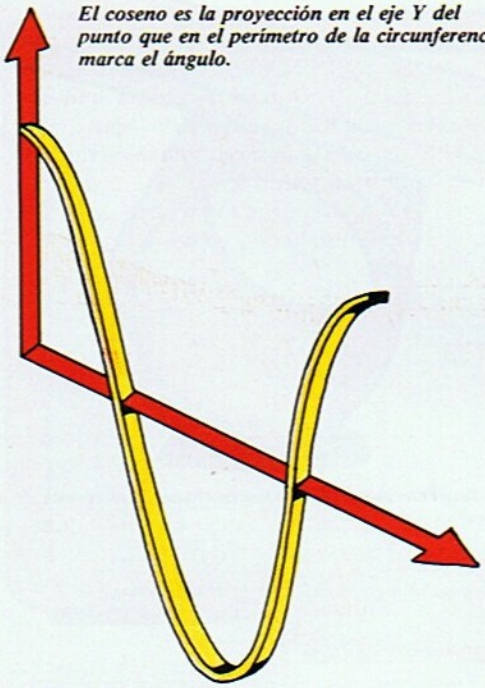
PRINT EXP 1

Como es lógico, los antilogaritmos que en el Spectrum se pueden obtener directamente, son los que toman como base el número **e**.
Para finalizar con las funciones logarítmicas, y sólo para aquellos que tengan la especial curiosidad, y los suficientes conocimientos, para comprender el sistema por el cual el Spectrum calcula los logaritmos, diremos que una parte de su ROM, más concretamente dentro de la zona denominada CALCULADOR, incorpora una rutina que genera la denominada serie de polinomios de Chebyshev, gracias a lo cual, calcula las aproximaciones a las funciones **SIN, ATN, LN** y **EXP**, y por tanto a sus derivadas (**COS, TAN, ASN, ↑** y **SQR**).

FUNCIONES TRIGONOMETRICAS

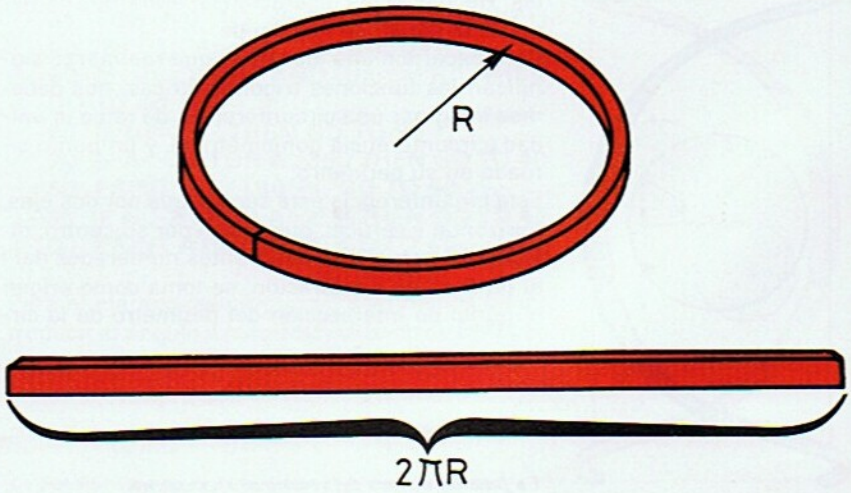
En este apartado, vamos a tratar tres funciones y sus correspondientes inversas. Los nombres de las funciones son: **SIN, COS, TAN** y **ASN, ACS, ATN;** estas se corresponden con SENO, COSENO, TANGENTE y ARCOSENO, ARCOCOSENO, ARCOTANGENTE. En todos los casos, la unidad angular que se ha de utilizar para expresar el argumento de las tres primeras funciones (no de sus inversas), es el RADIAN: arco cuya longitud es igual al radio de la circunferencia; por tanto, los ángulos posibles en una circunferencia, expresados en radianes, oscilan entre 0 y 2π. Como la mayoría de nosotros sabremos, PI es la constante matemática que relaciona la longitud de una circunferencia con su diámetro, y tiene el valor aproximado 3.1415927. Manejando RADIANES, se hacen muy frecuentes referencias a este valor, y el Spectrum lo tiene almacenado como una constante interna, de forma que puede ser empleada por el usuario como si se tratase de una variable de nombre PI, cuyo valor está predefinido y no es posible alterar. Así, por ejemplo, si deseamos conocer el valor que esta constante

El coseno es la proyección en el eje Y del punto que en el perímetro de la circunferencia marca el ángulo.



tiene en el ordenador, podemos escribir: **PRINT PI.**
Debemos tener mucho cuidado, para no confundir en el Spectrum la constante **PI** con una posible variable numérica PI; a efectos del ordenador, son absolutamente diferentes. La constante **PI**, se obtiene pulsando la tecla **M** en el modo extendido, y corresponde al carácter 167 en el código del Spectrum, mientras que la posible variable PI, se introduce carácter a carácter, y es equivalente a pi, Pi o pl.
Si al encender el ordenador, le pedimos que nos diga el valor de la variable PI, (**PRINT PI**) veremos que nos contesta con el error **2 Variable not**

La constante PI es la relación entre la longitud de una circunferencia y su diámetro; su valor aproximado es 3.1415927.

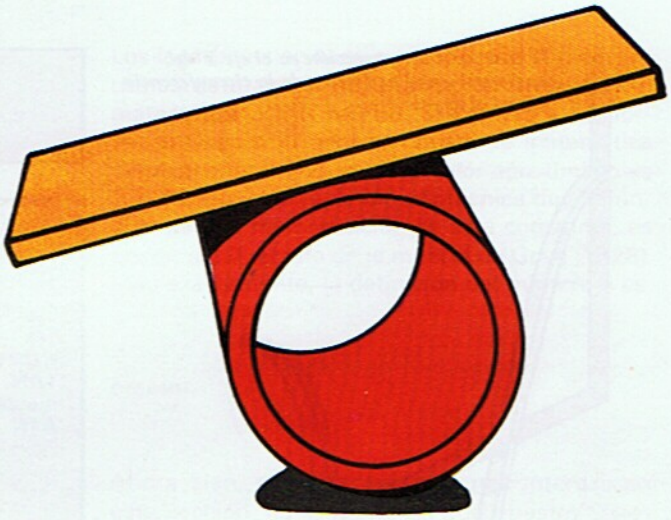


¡!

Las funciones inversas de las trigonometricas **SIN, COS** y **TAN**, son, respectivamente, **ASN, ACS** y **ATN**.

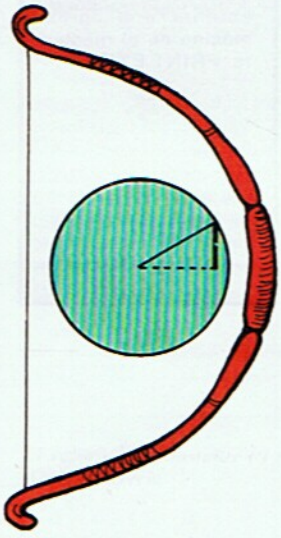
*

La base de los logaritmos neperianos es el número **e**. Podemos obtener una aproximación de él mediante: **PRINT EXP 1**.



La tangente es el cociente entre el seno y el coseno de un ángulo.

La función inversa del seno es el arcoseno.



found (variable no encontrada), mientras que si ejecutamos **PRINT PI**, siendo **PI** la constante (**EXTENDED MODE** y **M**), obtendremos el valor que ya conocemos.

Del mismo modo, si intentamos asignar un valor a la posible variable PI, observaremos que la tarea se lleva a cabo sin ninguna complicación, pero los valores obtenidos a continuación con **PRINT** PI (variable) y **PRINT PI** (constante), serán absolutamente diferentes (¡a no ser que hayamos ejecutado **LET** PI=**PI**!). Por otra parte, cualquier intento por alterar el valor de dicha constante, ya sea mediante **LET**, **INPUT** o cualquier otro sistema, será rechazado por el *syntax checker*.

Una buena idea, para evitar posibles confusiones en los listados, entre la variable PI y la constante **PI**, es escribir la variable siempre con minúsculas. Hechas estas aclaraciones, entremos de lleno en el tema que nos ocupa.

Para hacernos una idea de lo que realmente significan las funciones trigonométricas, nos debemos imaginar una circunferencia de radio la unidad (circunferencia goniométrica), y un punto situado en su perímetro.

Esta circunferencia está seccionada por dos ejes, horizontal y vertical, que pasan por su centro, dividiéndola en cuatro cuadrantes numerados del I al IV. Para su numeración, se toma como origen el punto de intersección del perímetro de la circunferencia con el extremo derecho del eje horizontal, considerando por convenio el sentido de desplazamiento positivo el antihorario, es decir, el contrario al de las agujas del reloj.

Del desplazamiento de este punto por el perímetro de la circunferencia, se deducen una serie de relaciones entre el ángulo descrito en el movimiento y las proyecciones sobre los ejes horizontal y vertical. Las proyecciones sobre el diámetro horizontal o eje X, corresponden a la representación de la función coseno, y las proyecciones sobre el diámetro vertical o eje Y, las del seno. Los valores del seno y del coseno en el primer cuadrante son positivos; en el segundo, el seno es positivo, mientras que el coseno es negativo; en el tercero, tanto el seno como el coseno resultan negativos; por último, en el cuarto cuadrante, el seno resulta negativo y el coseno positivo, con lo cual, a lo largo de la circunferencia, se producen todas las combinaciones posibles.

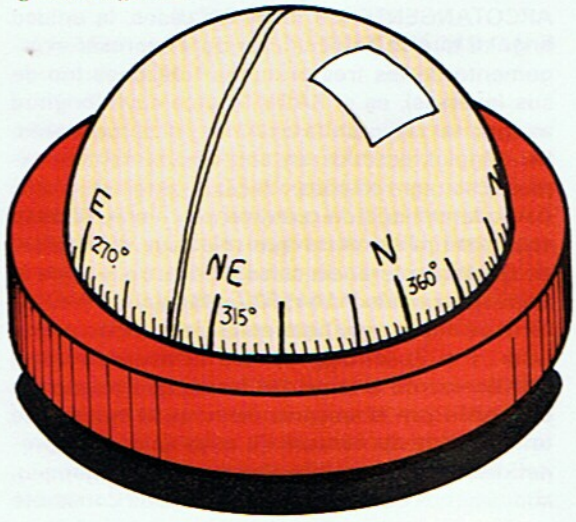
Otra propiedad interesante de las funciones angulares, que se nos hace evidente dada la estructura circular en que el imaginario punto del perímetro se desplaza, es que los valores de las funciones se repiten una vez cumplida una vuelta completa; por tanto, podemos afirmar que:

SIN X=SIN (X+N*PI) y COS X=COS (X+N*PI)

Siendo **X** el ángulo descrito, y **N** cualquier número entero múltiplo de dos, que expresa el doble del número de «vueltas» dadas a la circunferencia.

Existe una tercera función trigonométrica básica, que se obtiene como la relación existente entre el seno y el coseno de un ángulo. A esta relación se la conoce con el nombre de tangente, y pode-

Las dos unidades más frecuentemente utilizadas en la goniometría (medición de ángulos), son el radián y el grado sexagesimal.



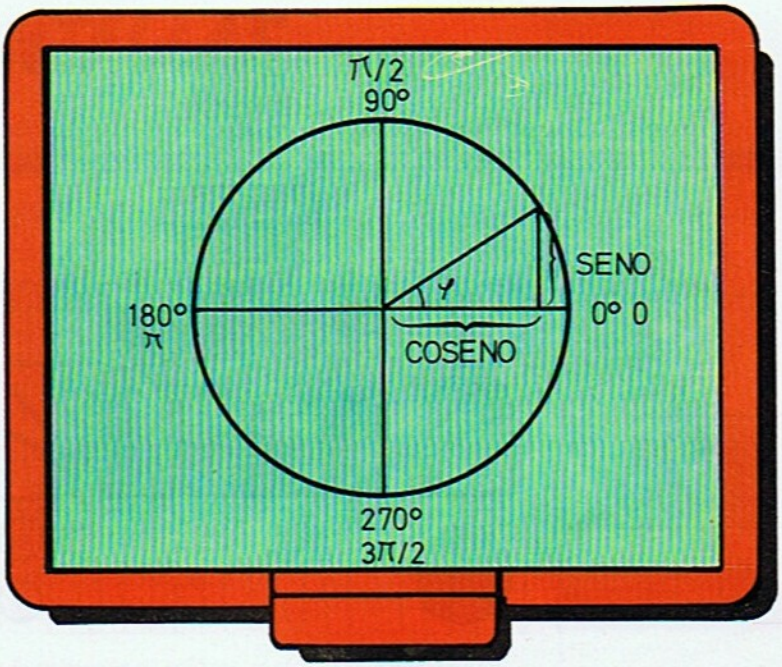
La función inversa del coseno es el arcoseno.

mos definirla como: **TAN X=SIN X/COS X.** Tanto la función seno como coseno, oscilan siempre entre los valores 0 y 1. Esto resulta obvio, si pensamos en que el valor mínimo que pueden adquirir es cero, como es el caso del seno de 0 o del coseno de $\pi/2$; y el valor máximo posible es 1, puesto que el radio de la circunferencia es la unidad, como es el caso del seno de $\pi/2$ o el coseno de 0.

De lo dicho anteriormente, podemos deducir que cualquier valor tendrá sentido como argumento de las funciones **SIN** y **COS,** pero no sucederá así con los argumentos de la función **TAN.** Dado que la función tangente es el cociente de las funciones seno y coseno, debemos preocuparnos de no emplear argumentos que al hacer cero el denominador, puedan llevarnos a un error del tipo **6 Number too big** (número demasiado grande). Esto se produce, lógicamente, con aquellos valores angulares cuyo coseno sea cero, es decir $\pi/2$ y $3/2\pi$.

Por otro lado, podemos disponer de las funciones inversas a las anteriormente citadas, que son el arcoseno **(ASN),** arcocoseno **(ACS),** y arcotangente **(ATN).** Estas funciones nos permiten obtener, a partir del valor de un seno, un coseno o una tangente, respectivamente, el ángulo (arco) del cual provienen.

Esto implica que los argumentos de las funciones trigonométricas inversas, deberán estar siempre comprendidos entre 0 y 1, puesto que, como hemos visto anteriormente, los valores entre los cuales se mueven los resultados de las funciones **SIN, COS** y **TAN** son éstos. Si pese a esta advertencia, intentamos efectuar una función inversa de este tipo, con argumento no comprendido entre 0 y 1, nos las veremos cara a cara con el mensaje **A Invalid argument** (argumento erróneo).



Una circunferencia goniométrica (de radio uno), nos facilita la labor de representar las relaciones trigonométricas.

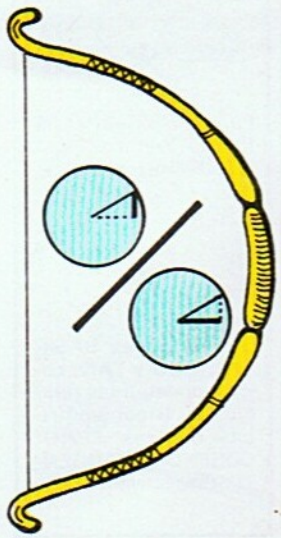
UNIDADES ANGULARES

Cuando comenzamos a hablar de las funciones trigonométricas, aseguramos que la unidad en que se miden las magnitudes angulares es el radián; sin embargo, si bien es cierto que esta es la unidad que más se maneja en medios técnicos, y que emplea el ordenador, la mayoría de nosotros utilizamos a tal fin el llamado sistema SEXAGESIMAL, en el cual la unidad es el grado sexagesimal, y sus submúltiplos: el minuto (60 minutos=1 grado) y el segundo (60 segundos=1 minuto).

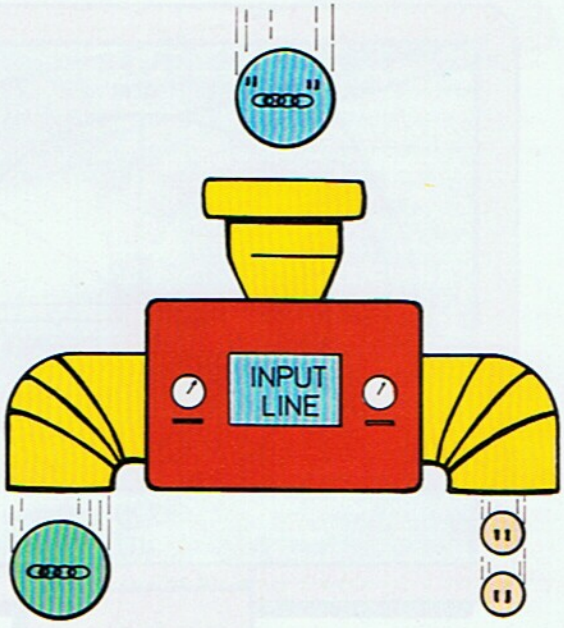
La conversión de un sistema a otro es muy sencilla. Sólo nos hace falta saber que el sistema sexagesimal divide la circunferencia en 360 grados; por tanto, existe una relación de 360 grados por cada 2 radianes, o lo que es lo mismo, π radianes equivalen a 180 grados. Así pues, para pasar radianes a grados, sólo hemos de multiplicarlos por 180 y dividirlos por π; y para la operación contraria, es decir, convertir grados en radianes, multiplicar los grados por π y dividirlos por 180. El siguiente programa nos muestra como efectuar la conversión de una forma práctica:

```
 10 REM CONVERSION ANGULAR - J.M.
 LOPEZ MARTINEZ
 20 PRINT "GRADOS","RADIANES"
 30 FOR I=0 TO 31:PRINT "-";:NEXT I
 40 INPUT "Angulo:";A,LINE U$
 50 IF U$<>"G" AND U$<>"R" THEN
 GO TO 40
 60 IF U$="R" THEN GO TO 90
 70 IF A<-360 OR A>360 THEN GO TO 40
 80 PRINT A,A/180*PI: GO TO 40
 90 IF A<-2*PI OR A>2*PI THEN GO TO 40
100 PRINT A/PI*180,A: GO TO 40
```

Para el manejo del programa, sólo hemos de introducir el ángulo a convertir (seguido de **ENTER**)



La función inversa de la tangente es el arcotangente.



INPUT LINE *suprime las comillas que acompañan la entrada de variables de cadena.*

y a continuación la unidad en que se encuentra; para ello, utilizaremos una **R** cuando sean radianes, y una **G** cuando sean grados sexagesimales. A este respecto, conviene advertir que dichas letras debemos entrarlas en mayúsculas, para que sean reconocidas por el programa (líneas 50 y 60).

En lo referente al programa, en su mayoría se utilizan elementos que ya conocemos: la línea 10 contiene el **REM** de nombre y autoría, la 20 y 30 realizan la impresión de una cabecera (GRADOS RADIANES) subrayada con guiones, las líneas 70 y 90 efectúan depuraciones de los valores introducidos y, finalmente, las líneas 80 y 100, llevan a cabo el cálculo e impresión de los resultados. Ahora bien, intencionadamente hemos dejado para el final la instrucción 40, puesto que en ella aplicamos a la entrada de datos **(INPUT)** algunos conceptos nuevos, que conviene analizar más detenidamente. En primer lugar, utilizaremos un **INPUT** en el que se visualizará el literal especificado, pero en vez de una sola variable, se producirá la entrada de dos; una numérica **(A)**, que indica la magnitud del ángulo a convertir, y otra de cadena **(U$)**, que portará la unidad de medida angular (GRADOS=G, RADIANES=R).

Observemos que la primera variable del **INPUT** será solicitada inmediatamente a continuación del mensaje de petición (para designar este tipo de mensajes se utiliza la palabra inglesa *PROMPT)*, mientras que la segunda lo hará en la columna dieciséis, debido al separador coma **(,)** que se ha introducido entre las dos variables. Para la entrada de la segunda variable **(U$)**, del tipo cadena, hemos utilizado el formato **INPUT LINE**, cuyo objetivo es la supresión de las comillas que en los **INPUT** acompañan a los datos de este tipo. De esto deducimos, que **INPUT LINE** sólo puede ser empleado con variables de cadena, y que tiene una finalidad meramente estética.

Por último, conviene recordar que para abandonar los **INPUT** de variables numéricas, empleábamos el comando **STOP**, y para los de variables de cadena, previamente a la realización de esta operación, procedíamos a la eliminación de las comillas, que señalan este tipo de datos, bien mediante **DELETE**, o bien mediante **EDIT**. Sin embargo, en el caso del **INPUT LINE** ninguno de los dos sistemas funcionará, puesto que ni la variable a entrar es numérica, ni aparecen comillas que poder suprimir.

Para detener los programas durante la ejecución de este tercer tipo de **INPUT**, deberemos pulsar la tecla CURSOR ABAJO (**CAPS SHIFT** + 6), lo que producirá la generación de un mensaje del tipo **H STOP in INPUT**.

Para producir la detención de un programa durante la ejecución de un INPUT LINE, *es necesario pulsar CURSOR ABAJO* (CAPS SHIFT + 6).

¡!

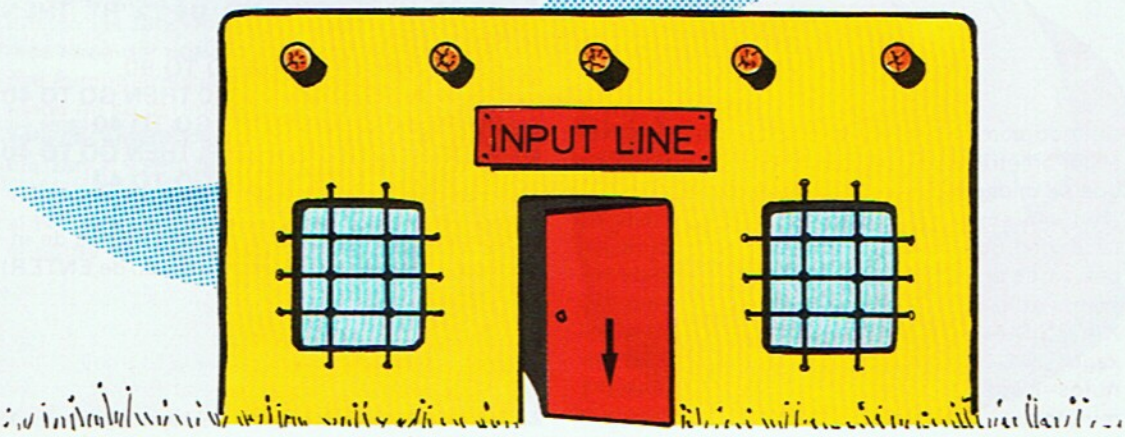
La función logarítmica (aplicada a neperianos) se obtiene en el Spectrum mediante **LN**.

*

La potenciación es la función con más prioridad dentro de las operaciones matemáticas básicas.

*

Las funciones BASIC **SIN**, **COS** y **TAN**, corresponden a las relaciones trigonométricas básicas SENO, COSENO y TANGENTE, respectivamente.

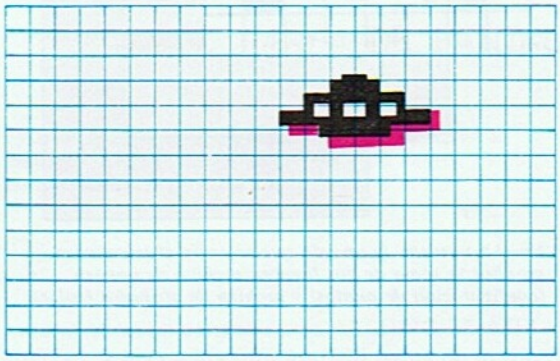


DINAMISMO DE PANTALLAS

COMO ya mencionamos anteriormente, la pantalla es el medio fundamental de comunicación con el usuario y, por tanto, a la hora de la realización de un programa, hemos de hacer hincapié en la importancia de la misma. Como vimos anteriormente, el tratamiento de la pantalla se apoya en tres puntos fundamentales:

— Diseño de la pantalla.
— Utilización del color.
— Dinamismo.

Pero ¿qué es el dinamismo? Caeremos en la tentación del chiste fácil, y diremos que no es sólo importante cuidar la estética de la pantalla, sino también su «estática»; esto debe pasar de ser un mero juego de palabras y convertirse en una máxima que nos acompañe durante todas nuestras tareas de programación.
El ordenador, al contrario que otros vehículos de información, como el cartel, la fotografía o el libro, dispone de la gran ventaja de ser dinámico, es decir, en la mano del programador está el alterar, a su gusto, el contenido informativo, y como no, la forma en que éste se presenta. Desaprovechar esta cualidad del ordenador, sería un error imperdonable. Encendamos el ordenador y veamos en un ejemplo el efecto que el dinamismo puede imprimir a la simple presentación de un mensaje:



En la animación de gráficos, se efectúa el desplazamiento de un sprite sobre un fondo fijo.

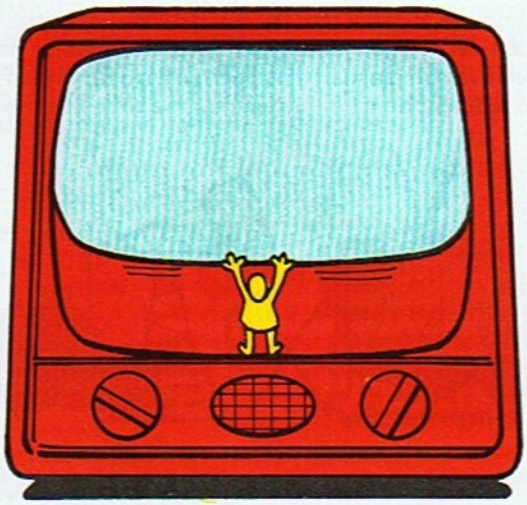
```
10 PRINT AT 11,7;"PULSA UNA TECLA"
20 IF INKEY$="" THEN GO TO 20
30 CLS
40 PRINT AT 11,7;OVER 1;"PULSA UNA
   TECLA"
50 BEEP .5,60
60 IF INKEY$="" THEN GO TO 40
```

Evidentemente, el objetivo de la presentación del mensaje (transmitir al usuario una breve información), ha sido cumplido más ampliamente en el segundo caso, puesto que, además de esa transmisión de información, se ha reclamado previamente la atención del destinatario de la misma.
Los efectos dinámicos son, por tanto, esenciales en la programación, y con vistas a facilitar su estudio pueden ser divididos en dos grandes grupos, según la magnitud de la zona de pantalla que se vea afectada por su acción:

— Dinamismo de pantalla.
— Dinamismo de gráficos.

En el primer caso, la zona sobre la que se aplica el efecto de dinamismo es de dimensiones considerables, generalmente toda la pantalla, aunque las técnicas utilizadas a tal fin pueden ser empleadas también en áreas más restringidas

El scroll es un ejemplo claro de dinamismo a pantalla completa, puesto que toda ella (fondo y primer término), es desplazada de manera solidaria.



¡!

Los bits dentro de un byte se numeran del 0 al 7, a partir de la derecha. El término «significativo» determina la importancia que un bit tiene en el valor de un byte. Así pues, los bits serán tanto más significativos cuanto más a la izquierda se encuentren, es decir, cuanto mayor sea la posición que ocupan.

*

Debido a la disposición de los bits dentro de los bytes del área de atributos, el valor decimal de uno de estos bytes se puede obtener por la siguiente fórmula: valor = código de tinta + código de fondo * 8 + brillo * 64 + intermitencia * 128.

¡!

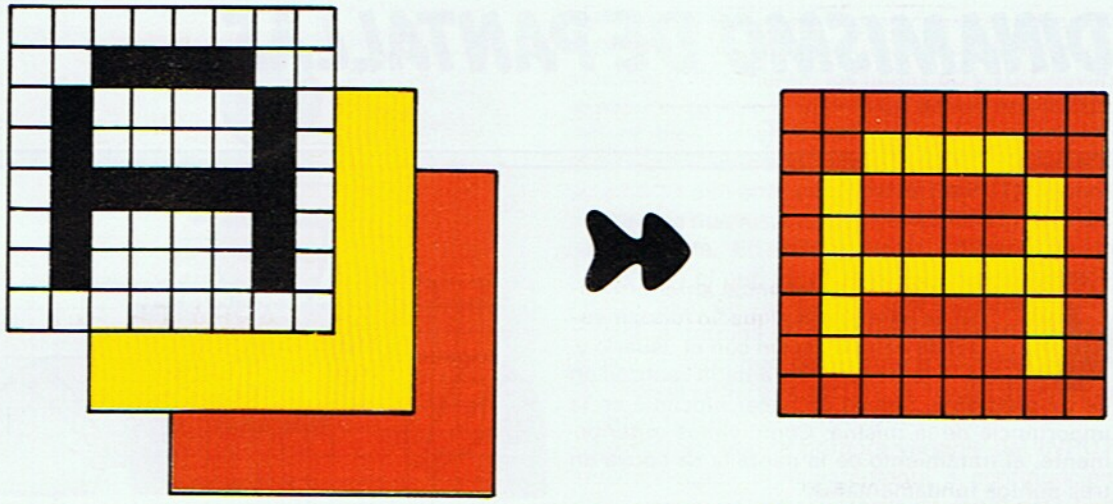
Cada bit del archivo de imagen corresponde a un punto de la pantalla del Spectrum. Puesto que cada uno de ellos es controlable independientemente, se dice que la pantalla de nuestro ordenador es DIRECCIONABLE PUNTO A PUNTO.

*

A la mínima unidad gráfica que en un ordenador se puede representar se le denomina *pixel*, abreviatura de la palabra inglesa *picture element*, cuyo significado es ELEMENTO GRAFICO.

*

La cantidad de *pixels* que se pueden representar simultáneamente en un ordenador, miden la RESOLUCION de la pantalla del mismo. En el caso del Spectrum, se pueden representar 49.152 *pixels*, por lo que se le considera de alta resolución.



En el Spectrum, la imagen que se envía al televisor o monitor se toma de dos fuentes: archivo de imagen (configuración de puntos) y archivo de atributos (color).

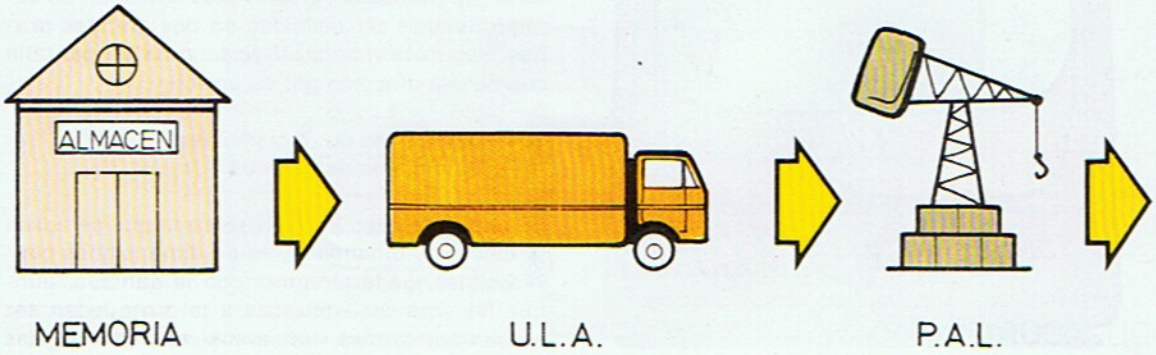
como, por ejemplo, un tercio o dos tercios de la pantalla. En todo caso, podemos decir que las zonas afectadas, constituyen generalmente rectángulos con el marco de la pantalla, y el efecto de dinamismo se distingue particularmente porque se aplica sobre el área completa que afecta, incluyendo tanto el fondo como el primer término, que puede estar constituido por una o varias unidades gráficas (texto, gráficos, etc...).

El segundo tipo de dinamismo, se distingue del anterior precisamente porque el objeto del mismo es la animación de determinadas unidades gráficas, generalmente reducidas en tamaño, y sobre las cuales se actúa independientemente del fondo; por ende, los sujetos pasivos de su acción, que en el argot microinformático se conocen como *sprites* (unidades gráficas), no suelen ser superficies regulares, debido a lo cual las técnicas a emplear son absolutamente distintas de las utilizadas para el dinamismo a pantalla completa.

Finalmente, podemos hablar del dinamismo en aplicaciones muy concretas, como es por ejemplo la presentación de mensajes, de extraordinaria importancia debido a su frecuente uso, y que disfruta del beneficio de ciertos trucos y técnicas especiales, obtenidos por extensión de los que se emplean para la animación de gráficos, y que se benefician de la mayor facilidad de tratamiento que el lenguaje BASIC aporta para los textos. Un último punto a destacar en el tema del dinamismo, es la aplicación del sonido a este fin; sin embargo, éste merece un tratamiento independiente, por lo que será excluido de las técnicas de animación general que estudiaremos en las próximas páginas.

Aunque sin duda estaremos deseosos de adentrarnos en las técnicas de dinamismo, con las cuales podremos obtener espectaculares efectos, deberemos frenar nuestro ímpetu y adquirir previamente unas ciertas ideas básicas sobre la forma en que nuestro Spectrum trata la información que se envía al televisor o monitor. Estos conocimientos nos ayudarán a comprender mejor algunas técnicas y trucos, que debido a la utilización del código máquina y otras herramientas de relativa complejidad, nos podrían resultar un tanto arduas o difícilmente asimilables sin esta base.

El proceso de emisión de imagen en el Spectrum es llevado a cabo por varios elementos de su hardware.



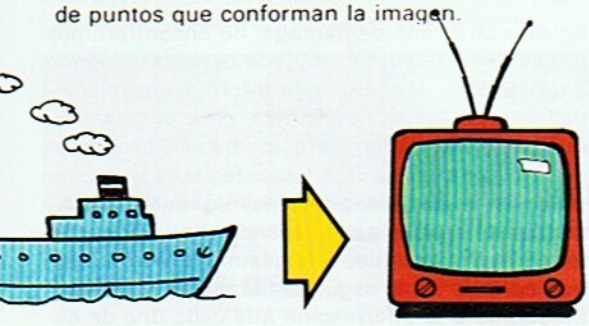
LA PANTALLA EN LA MEMORIA

La emisión de imagen en el Spectrum es un proceso de relativa complejidad, que involucra varios elementos del ordenador: la U.L.A. se encarga de recoger la información a transmitir de la memoria R.A.M.; desde aquí, los datos son enviados al codificador P.A.L. y posteriormente al modulador de U.H.F., que se ocupa de preparar la señal para que sea recibida por el televisor a través del canal 36 de U.H.F. (este último paso se ahorra al utilizar un monitor).

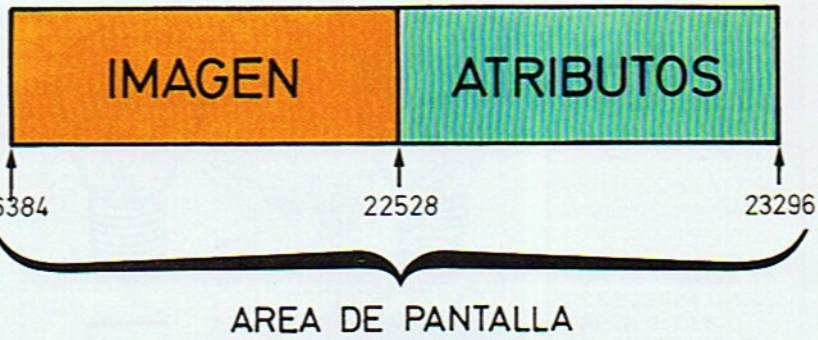
Como hemos visto, muy a grandes rasgos, el proceso de emisión de imagen es automático y concierne directamente a dispositivos físicos del aparato *(hardware)*, gracias a lo cual, nuestra tarea de programación se limitará única y exclusivamente a depositar en la zona de memoria adecuada, los datos que se desean transferir a la pantalla; este área de la memoria se denomina MEMORIA DE PANTALLA, y ocupa un total de 6.912 bytes, comprendidos desde la dirección 16.384 (primera de la memoria R.A.M.) hasta la 23295, ambas inclusive.

Para cualquier profano en la microinformática, la imagen transmitida por el ordenador es una determinada forma multicolor; sin embargo, para el Spectrum, y por ende para el programador, la imagen se divide en dos partes claramente diferenciadas: por un lado, los puntos que la integran, y por otro el color de éstos. Del mismo modo, la zona de memoria de pantalla se divide en dos áreas claramente distintas, destinadas cada una de ellas a contener uno de estos elementos de la imagen (figura y color).

La primera de ellas se denomina ARCHIVO DE IMAGEN, y ocupa 6.144 bytes (desde la dirección 16384 hasta la 22527, ambas inclusive). Esta zona se destina al almacenamiento de la configuración de la pantalla, es decir, la distribución de puntos que conforman la imagen.



MODULADOR



El área de memoria dedicada al almacenamiento de la pantalla, se divide en dos zonas: una para imagen y otra para atributos.

La otra zona se conoce como ARCHIVO DE ATRIBUTOS, y ocupa los últimos 768 bytes de la memoria de pantalla (desde 22528 hasta 23295), siendo su misión almacenar la información referente al color de la pantalla.

EL ARCHIVO DE IMAGEN

Cada uno de los puntos que configuran la imagen en la pantalla, puede estar «encendido» o «apagado» (color de primer término o color de fondo); dado que esta circunstancia supone únicamente dos posibles estados, cabe su representación informática mediante un bit, en el que se adopta el convenio 1-ENCENDIDO, 0-APAGADO.

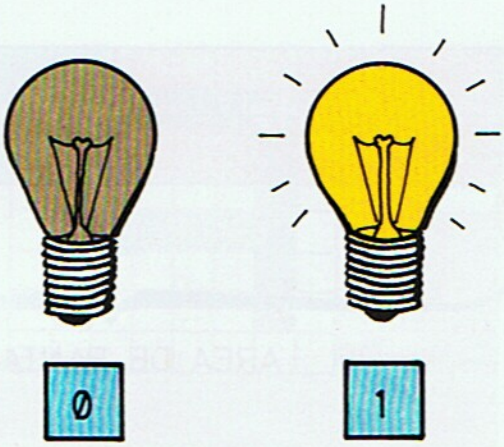
La imagen en el Spectrum está constituida por un total de 49.152 puntos, distribuidos en 192 líneas de 256 puntos cada una. Puesto que cada 8 bits suponen un byte de información, los puntos horizontales se agrupan de ocho en ocho, dando lugar a un archivo de imagen compuesto por 192 líneas de 32 bytes cada una (6.144 bytes).

Por tanto, para conectar determinado punto de la pantalla, sólo tendremos que depositar la información binaria conveniente en el byte del archivo de imagen que deseemos. Así, por ejemplo, efectuando **POKE 16384,255** conectaremos los ocho primeros puntos horizontales de la pantalla, puesto que la configuración binaria de 255 es 11111111; si sólo deseamos conectar un punto, no tendremos más que buscar la configuración

¡!

Todas las líneas que componen un programa escrito en BASIC, deben comenzar por un número comprendido entre 1 y 9999. Por esto, al ejecutar, por ejemplo, LIST 20000, la pantalla en blanco y el informe OK, corroboran la imposibilidad de encontrar tal línea. Sin embargo, al efectuar LIST 49162, obtenemos el mismo resultado que con LIST 10, es decir, el ciclo del argumento del comando LIST se repite a partir de 49152. Idéntica particularidad se cumple con LLIST. Probemos el siguiente programa:

```
10 INPUT "LISTADO
A PARTIR DE ";X
20 CLS: PRINT
"LIST ";X: PRINT
30 LIST X : GOTO 10
```



BIT

Cada bit del archivo de imagen representa en la pantalla un punto encendido o apagado.

binaria adecuada; por ejemplo: **POKE 16384,1** conecta el octavo punto de la pantalla (1 decimal = 00000001 binario), **POKE 16384,128** el primero (128 decimal = 10000000 binario), o **POKE 16384,197** los puntos 1, 3, 7 y 8.

Habremos observado claramente, que la modificación del contenido de la dirección 16.384 (primera del archivo de imagen), altera lo que sería la primera línea del primer carácter de la primera fila; de forma similar, 16.385 controla la primera línea del segundo carácter de la primera fila. Hasta este punto todo es muy fácil de retener, ya que cada byte se corresponde con una columna de caracteres; sin embargo, según este razonamiento, el byte 33 del archivo de imagen, debería corresponder a la segunda línea del primer carácter de la primera fila, y esto lamentablemente no se cumple: este byte es el que controla la primera línea del primer carácter de la segunda fila.

Así pues, las complicaciones aumentan, ya que en la memoria no se ordenan los bytes tal y como luego aparecen en la pantalla del televisor, sino que primeramente se encuentran las primeras líneas de todas las filas de caracteres, a continuación las segundas, las terceras, etc. hasta llegar a las octavas (últimas).

Lamentablemente, las desgracias nunca vienen solas, y esta distribución tampoco es del todo cierta. A efectos de estructura de la memoria, la pantalla se divide en tres tercios de ocho filas cada uno (64 líneas). Cada uno de ellos ocupa, por tanto, 2 K de memoria (2 × 1.024 = 2.048 bytes), lo que suma el total de 6 K del archivo de imagen (6 × 1.024 = 6.144 bytes).

Así pues, la exacta distribución de este archivo es: primeras líneas de las ocho primeras filas, segundas líneas de las ocho primeras filas, etc. hasta las octavas líneas de las ocho primeras filas; a continuación, se representan las ocho primeras líneas de las filas 9 a 15, y unas vez finalizado este tercio, es decir, representadas las ocho octavas líneas de la fila 9 a la 15, se pasa a la representación del último tercio, siguiendo el mismo sistema. Bueno, ¡por lo menos los tres tercios si que están por orden!

Esta organización, aparentemente un tanto caótica, se debe a razones de *hardware*, que en nuestro caso es como decir «razones de peso». En todo caso, la distribución de la pantalla es algo que deberemos tener muy presente al experimentar con las técnicas de dinamismo a pantalla completa. Por otra parte, esta es la explicación de por qué las pantallas que el Spectrum almacena, se cargan de una manera un tanto peculiar.

Para dar un repaso a como se estructura el archivo de imagen en nuestro ordenador, ejecutemos el siguiente programa, en el que la pantalla se irá llenando de líneas según el orden de los byes que las controlan:

```
10 PAPER 7:INK 0: BORDER 6: CLS
20 FOR I=0 TO 21
30 PRINT I
40 NEXT I
50 FOR I=16384 TO 22527
60 POKE I ,255
70 NEXT I
80 GO TO 80
```

Como habremos podido observar, la diferenciación que desde el punto de vista de la programación se hace entre la zona de edición (inicialmente, las dos últimas líneas) y la de pantalla, no tiene ningún efecto en el archivo de imagen, que considera la pantalla como un todo único.

EL ARCHIVO DE ATRIBUTOS

En esta otra zona de pantalla, no encontraremos problemas de estructura. Cada byte corresponde a un carácter, y contiene la información relacionada con el mismo en lo referente al color. Su distibución está bien clara: los bytes se reparten de la misma forma que los caracteres que controlan en la pantalla; por tanto, el primer byte del archivo de atributos corresponde al primer carácter de la primera fila, el trigésimo tercero al primer carácter de la segunda fila, etc.

En cuanto a la información que cada uno de estos bytes porta, se distribuye de la siguiente manera: los tres bits menos significativos (los de

¡!

Cuando realizamos la impresión mediante el tabulador **TAB**, durante el salto de una columna a otra, el ordenador memoriza el color en curso en ese momento. El siguiente programa lo demuestra:

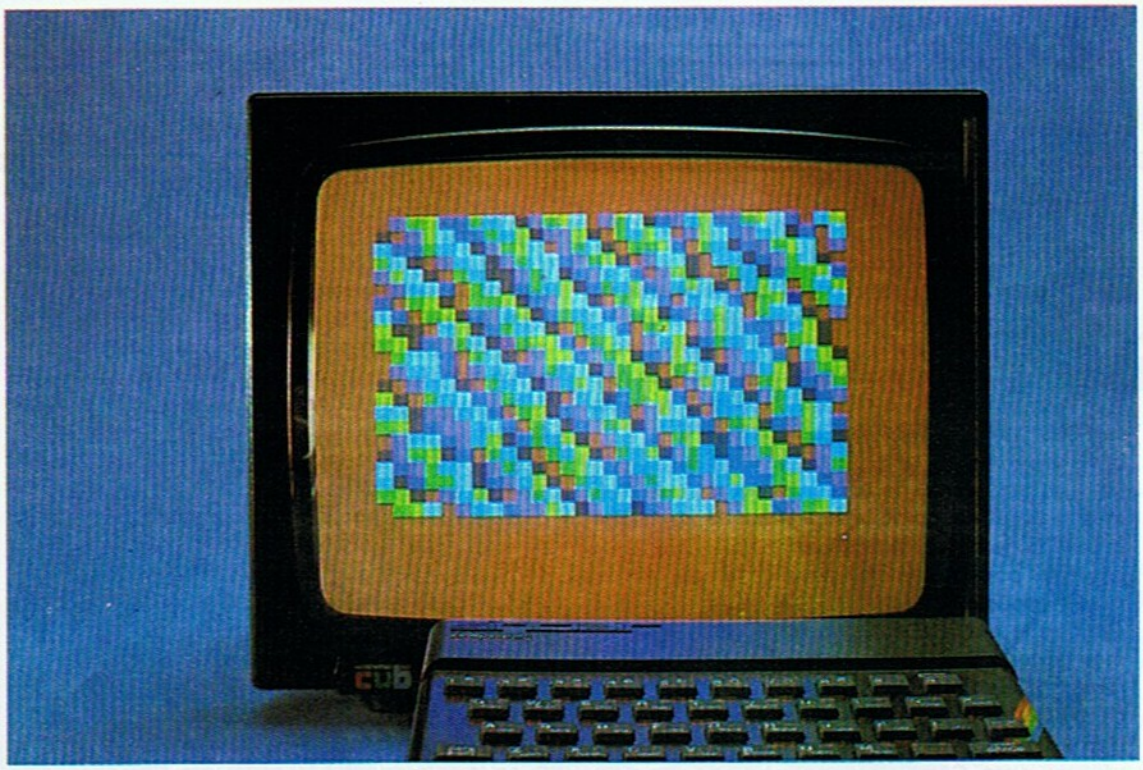
```
10 FOR A=0 TO 20
20 PRINT PAPER 7; TAB 0;A; PAPER RND*6.5;TAB 15;A; PAPER 7; A;PAPER RND*6.5;TAB 29; PAPER 7;A
30 NEXT A
```

Si cambiamos los **TAB** anteriores por **AT**, veremos que cuando utilizamos este último no sucede lo mismo.

*****

Cuando realizamos cálculos matemáticos con nuestro Spectrum, muchos resultados se presentan con gran número de decimales, a veces, innecesarios. Si queremos limitar el número de estos, podemos incluir como subrutina en nuestros programas, la línea 20 del listado siguiente:

```
10 INPUT "CUANTOS DECIMALES ";D,"NUMERO A REPRESENTAR";N
20 LET A= INT (N*10↑D)/10↑D
30 PRINT A: GOTO 10
```



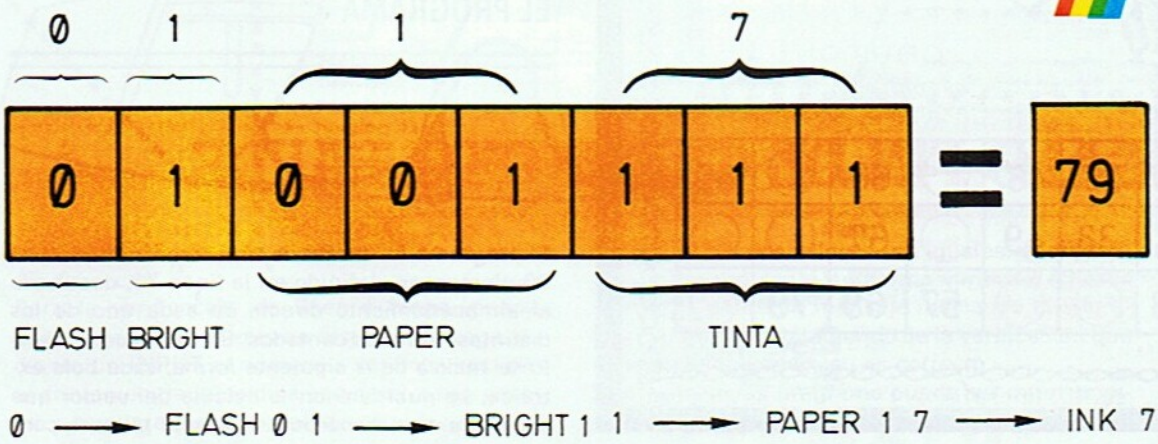
La organización del archivo de imagen divide la pantalla en tres tercios consecutivos, cada uno de los cuales ocupa 2 K. Cada uno de estos tercios, se encuentra a su vez compuesto por 64 líneas de 32 bytes.

más a la derecha), codifican el color de la tinta, los tres siguientes (bits 3, 4 y 5) el color del papel, el sexto bit controla el brillo, y el más significativo el **FLASH**.

En el siguiente programa escribiremos un asterisco en la pantalla, y veremos como mediante **POKE**s al área de atributos, podemos alterar sus características de color, sin cambiar su configuración de puntos. Si queremos aumentar la velocidad a la que discurre el programa, pulsemos cualquier tecla:

```
  10 CLS : PRINT "*": LET F=0
  20 PRINT '''''FLASH.:'''''BRIGHT:'''
"PAPER.:'''''INK....:"
  30 FOR I=0 TO 255
  40 POKE 22528,I
  50 LET C=I
  60 IF C>127 THEN LET F=1: LET C=C-128
  70 LET B=0: IF C>63 THEN LET B=1: LET
C=C-64
  80 LET P=INT (C/8)
  90 LET T=C-8*P
 100 PRINT AT 4,7;F;AT 5,7;B;AT 6,7;P;AT
7,7;T
 110 PAUSE 25
 120 NEXT I
```

Cada byte del área de atributos porta la información sobre tinta, fondo, brillo e intermitencia de un carácter de la pantalla.



¡!

Los aficionados a las bromas, encontrarán útiles, sin duda, las siguientes líneas:

```
10 CLS: PAUSE 15:
POKE 23624,0: CLS:
PAPER 0: CLS
15 PAUSE 55: PA-
PER 7: POKE
23624,56: CLS
20 PRINT #1;"©
1982 Sinclair Re-
search Ltd": PAUSE
0
```

*

En la foto de la izquierda, aparece una pantalla muy alegre que demuestra el efecto de **POKE** sobre el área de atributos. He aquí el programa que la genera: **10 POKE 22528+RND*768, RND*255: GOTO 10**

¡BINGO!

EL juego del bingo o lotería casera, ha alcanzado con el paso del tiempo un gran número de adeptos. Así como el palé, el parchís o los juegos de cartas, el bingo es uno de los pasatiempos más socorridos para amenizar tardes aburridas.

El programa que a continuación estudiamos, gestiona la extracción de bolas de una forma totalmente aleatoria, como si se tratase de un perfecto bombo de bingo, memorizando cada uno de los números para la posterior verificación de línea o bingo respectivamente.

Aunque la mayoría de nosotros conoce la mecánica general de este juego, daremos un pequeño repaso a las reglas fundamentales: el elemento central es un bombo, dentro del cual se encuentran noventa bolas numeradas desde la uno hasta la noventa. Cada jugador dispone de uno o varios cartones con números, de forma que según se van extrayendo las bolas del bombo, se puede ir señalando en los mismos qué números de los presentes en nuestras tarjetas han aparecido. El jugador que primero tenga la suerte de tachar los cinco números de una misma línea de su cartón, será agraciado con el premio de LINEA (si esto ocurriera en una misma jugada con otros cartones, se repartiría el premio entre los afortunados). A partir de este momento, el premio de línea ha sido repartido, y ningún beneficio se otorgará a los próximos jugadores que obtengan línea. El siguiente premio, y el más codiciado, es el de BINGO, destinado a aquel o aquellos jugadores que antes tachen todos los números de su cartón.

El Spectrum no genera los cartones, debido a lo cual tendremos que conseguirlos de algún antiguo juego de bingo perdido por casa, o en el peor de los casos, fabricarlos nosotros mismos. Para ello debemos seguir determinadas normas:

— Los cartones se agrupan en series de seis, que contienen el total de bolas del bombo.

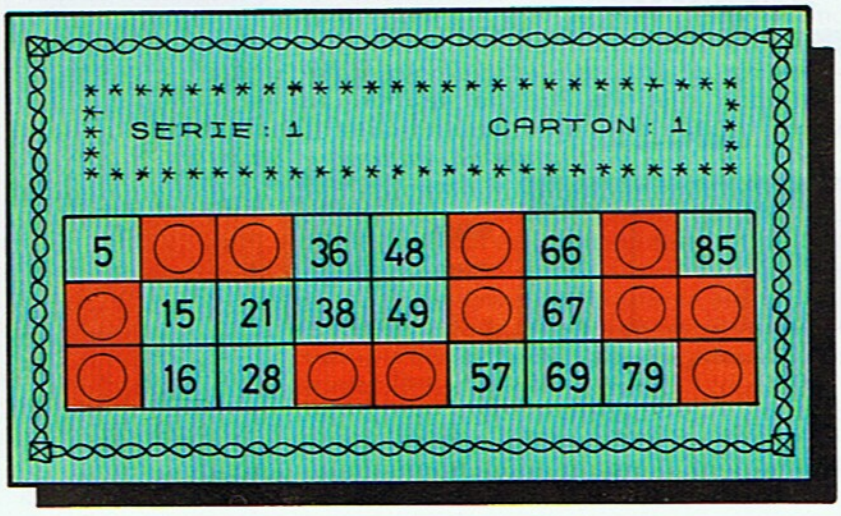
— Cada cartón presenta tres filas divididas en nueve columnas, en cada una de las cuales se colocan los números del cartón correspondientes a una decena: en la primera, los números del 1 al 9; en la segunda del 10 al 19, etc. Con excepción del 90, que se añadirá en la última columna, junto con las decenas de ocho.

— En cada casilla sólo habrá un número, lo cual implica que no podrá existir más de un número perteneciente a una determinada decena en una misma línea.

— Cada columna del cartón contendrá al menos un número, y nunca tres.

En cuanto a los premios, sugerimos que la línea se pague a una tercera parte del total de cartones recaudados, y el bingo con las dos terceras partes restantes; pudiéndose utilizar como moneda, el popular «dólar lentejiense», el «petro-garbanzo» o cualquier otra moneda admitida internacionalmente, para este tipo de juegos de azar.

Si carecemos de cartones de bingo o de alguna lotería casera, podemos fabricarlos nosotros mismos.



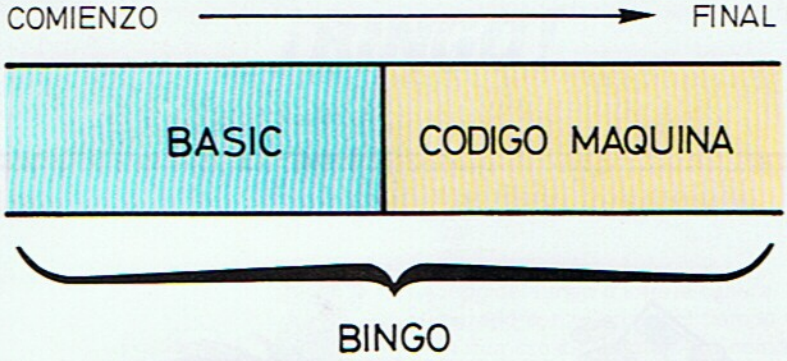
EL PROGRAMA

El programa hace uso de un vector numérico de 90 elementos, definido en la línea 70, destinado al almacenamiento directo de cada uno de los distintos números cantados. Este almacenamiento se realiza de la siguiente forma: cada bola extraída, se guardará en la celdilla del vector que corresponda a dicha bola. Así, si el número can-

tado es el 43, éste se almacenará en la casilla 43 del vector.

Para la verificación de línea o bingo, nuestro Spectrum utiliza el mismo sistema. Recoge mediante un **INPUT** el número a verificar, recupera el contenido de la celdilla asignada a ese número, y compara dicho contenido con el valor introducido. Si los dos valores son iguales, el ordenador pide el siguiente número a verificar; en caso contrario, emite un mensaje indicando que la línea o el bingo, dependiendo de la verificación que estemos efectuando, no es correcto.

La ejecución del programa puede ser interrumpida en cualquier momento, pulsando la tecla «B»,



El programa BINGO está compuesto por un soporte BASIC y una subrutina en código máquina, que deberán quedar almacenados en la cinta, según el orden que se indica en el gráfico.

lo que dará acceso a un menú de tres opciones: comenzar un nuevo bingo, verificar la línea cantada y verificar el bingo cantado.

En este programa, hemos utilizado nuevamente la subrutina en código máquina de caracteres gigantes (PSION Computers), por lo cual, aquellos que deseemos disfrutar del presente programa de bingo, deberemos introducir la citada subrutina aparecida anteriormente (programa LA BOMBA); bien a través del listado, siguiendo las instrucciones que a tal fin se brindaron en el comentario del programa mencionado, o bien grabándolo a continuación del soporte BASIC, si es que ya fue introducido en aquel momento.

La auto-ejecución del programa es obligatoria, por tanto, para la grabación del programa, tendremos que emplear el comando **SAVE "BINGO" LINE 1030.**

```
  10 REM ***********************          J.M.MAYORAL SERRANO          *******
***************
  20 POKE 23609,100: LET conta=1: INK 9: CLS : LET contador=0
  30 GO TO 260
  40 CLS : BORDER 0: PAPER 0: CLS : LET p$=" COMENZAMOS ": INK 1: PAPER 6: LET y
y=90: LET xs=2: LET ys=4: GO SUB 690: PAPER 0: LET ca=1
  50 FOR N=1 TO 10: BEEP .2,20: BEEP .2,5: NEXT N
  60 POKE 23609,100: LET conta=1: INK 9: CLS : LET contador=0
  70 CLS : DIM B(90)
  80 BORDER 0: PAPER 0: INK 9: CLS
  85 RANDOMIZE
  90 LET n=(INT (RND*90))+1
 100 IF B(n)=n THEN  GO TO 90
 110 LET B(n)=n
 120 LET CONTADOR=CONTADOR+1
 130 PAPER 0: CLS : BEEP .1,30: BEEP .15,40
 140 PLOT 81,50: DRAW 95,0: DRAW 0,77: DRAW -95,0: DRAW 0,-77
 150 PLOT 81,127: DRAW 20,10: DRAW 95,0: DRAW 0,-77: DRAW -20,-10
 160 PLOT 81+95,127: DRAW 20,10
 170 LET p$=STR$ B(n): LET ca=ca+1: GO SUB 680
 180 LET p$=STR$ contador: LET yy=5: LET xs=3: LET ys=3: LET xx=5: INK 6: PAPER
0: GO SUB 0700
 190 LET p$=STR$ contador: LET yy=150: LET xs=3: LET ys=3: LET xx=200: INK 6: PA
PER 0: GO SUB 700
 200 PAUSE 60
 210 IF INKEY$="b" THEN  INK 9: GO TO 260
 220 IF ca=91 THEN  GO TO 240
 230 GO TO 90
 240 PAPER 0: CLS : LET xs=1: LET ys=3: LET yy=100: LET p$=" SE ACABARON LAS BOL
AS ": PAPER 2: GO SUB 690: LET yy=150: LET ys=2: LET p$="Pulse una tecla para em
pezar": PAPER 0: GO SUB 690: PAUSE 0: RUN 40
 250 REM ***********************          FIN EJECUCION BINGO          *******
***************
 260 PAPER 1: BORDER 1: CLS : PRINT TAB 12;"OPCIONES"
 270 PRINT  PAPER 7; FLASH 1;AT 2,2;"1": PRINT AT 2,5;"EJECUCION DE BINGO"
 280 PRINT  PAPER 7; FLASH 1;AT 4,2;"2": PRINT AT 4,5;"VERIFICACION DE LINEA"
 290 PRINT  PAPER 7; FLASH 1;AT 6,2;"3": PRINT AT 6,5;"VERIFICACION DE BINGO"
 300 PRINT  PAPER 2;AT 10,0;"  Al cantar linea o bingo"; INVERSE 1;" pulsar  la
letra B
 310 PLOT 0,50: DRAW 255,0:            PLOT 0,40: DRAW 255,0
 320 FOR f=17 TO 21: FOR c=0 TO 31: PRINT  INK 6;AT f,c;"A": NEXT c: NEXT f
 330 PRINT  INK 9;AT 19,3;"INTRODUZCA NUMERO DE OPCION"
 340 LET K$=INKEY$
 350 IF K$="" THEN  GO TO 340
 360 IF K$="1" THEN : BEEP .2,40: LET conta=1: GO TO 40
 370 IF K$="2" AND CONTADOR>=5 THEN  BEEP .2,40: LET conta=1: GO TO 460
 380 IF K$="3" AND contador>14 THEN  BEEP .2,40: LET conta=1: GO TO 560
 385 IF contador<15 THEN  GO TO 340
 390 IF conta>=3 THEN  GO TO 1050
 400 CLS : IF conta>=2 THEN  LET conta=conta+1: LET p$="Mira que eres BRUTO": LE
T yy=60: LET xs=1: LET ys=2: GO SUB 690: GO TO 420
 410 GO TO 430
 420 LET xs=1: LET ys=2: LET yy=80: LET p$="Si solo tienes 3 opciones,": GO SUB
690: LET yy=100: LET p$="QUE PORRAS HACES PULSANDO UN ": GO SUB 690: LET yy=120:
LET p$="NUMERO MAYOR DE 3 ?!!!": GO SUB 690: PAUSE 200: GO TO 260
 430 INK 9: CLS : LET p$="POR FAVOR!": LET xs=2: LET ys=3: LET yy=50: GO SUB 690
: LET p$="Sabes leer?": LET yy=80: LET xs=2: LET ys=2: GO SUB 690: BEEP 3,-10
 440 LET conta=conta+1
 450 CLS : LET p$="Solo hay 3 opciones": LET yy=80: LET xs=1: LET ys=2: GO SUB 6
90: LET yy=110: LET p$="FIJATE BIEN": GO SUB 690: LET yy=140: LET p$="No creo qu
e sea tan dificil": GO SUB 690: BEEP 3,-20: GO TO 260
 460 LET cont=0
 470 CLS : PRINT AT 10,2;"INTRODUCIR UN SOLO NUMERO EN    CADA SECUENCIA"
 480 INPUT "Numero=";l
 485 IF L<=0 OR L>90 OR L-INT L THEN  BEEP 1,-10: GO TO 480
 490 IF B(l)=1 THEN  LET cont=cont+1: GO TO 540
 500 CLS : LET p$="EL": LET yy=10: LET xs=2: LET ys=4: GO SUB 690: LET p$=STR$ l
: LET yy=50: LET xs=5: LET ys=5: INK 5: BRIGHT 1: GO SUB 690: FLASH 0
 510 PAPER 1: INK 9: LET p$="NO LO HE DICHO": LET yy=100: LET xs=2: LET ys=3: GO
SUB 690: PAPER 0
 520 BRIGHT 0: BEEP .5,20: BEEP .5,30: PAUSE 100: PAPER 6: BORDER 6: INK 2: CLS
: LET p$="La linea no es correcta": BRIGHT 1: LET yy=70: LET xs=1: LET ys=3: GO
SUB 690: BRIGHT 0: PAPER 0:
 530 BEEP .5,30: PAUSE 150: CLS : GO TO 80
 540 INK 9: IF CONT=5 THEN  CLS : LET p$="LINEA CORRECTA": LET yy=80: LET xs=2:
LET ys=3: GO SUB 690: BEEP 2,50: PAUSE 100: CLS : GO TO 80
 550 GO TO 470
 560 LET cont=0
 570 CLS : PRINT AT 10,2;"INTRODUCIR UN SOLO NUMERO EN   CADA SECUENCIA"
 580 INPUT "Numero=";l
 590 IF B(l)=1 THEN  LET cont=cont+1: GO TO 650
 600 CLS : LET p$="EL": LET yy=10: LET xs=2: LET ys=3: GO SUB 690: PAPER 7: INK
2
 610 LET p$=STR$ l: LET yy=50: LET xs=5: LET ys=5: PAPER 1: INK 5: GO SUB 690: P
APER 1: INK 9
 620 LET p$="NO LO HE DICHO": LET yy=110: LET xs=2: LET ys=3: GO SUB 690: BEEP .
2,40: BEEP .2,10: PAUSE 100
 630 CLS : LET p$="El bingo no es correcto": LET yy=90: LET xs=1: LET ys=3: INK
6: BRIGHT 1: GO SUB 690: BRIGHT 0
 640 BEEP .2,40: BEEP .2,10:: PAUSE 150: PAPER 0: INK 9: CLS : GO TO 80
 650 INK 9: IF cont=15 THEN  CLS : LET P$=" BINGO CORRECTO": LET YY=80: LET XS=2
: LET YS=3: GO SUB 690: BEEP 2,50: PAUSE 100: CLS : GO TO 670
 660 GO TO 570
 670 RUN
 680 INK 7: BRIGHT 1: PAPER 2: LET yy=70: LET xs=5: LET ys=5
 690 LET xx=(256-8*xs*LEN p$)/2
 700 LET i=23306
 710 POKE i,xx: POKE i+1,yy: POKE i+2,xs: POKE i+3,ys: POKE i+4,8
 720 LET i=i+4: LET w=LEN p$
 730 FOR q=1 TO w
 740 POKE i+q,CODE p$(q)
 750 NEXT q
 760 POKE i+w+1,255
 770 LET w=USR 32256
 780 RETURN
 790 BORDER 7: PAPER 7:
 800 CLS
 810 INK 3
 820 LET p$="BINGO": BRIGHT 1: LET yy=0: LET xs=6: LET ys=10
 830 GO SUB 690
 840 LET p$="ZX.Spectrum": BRIGHT 0: LET yy=80: LET xs=2: LET ys=4: INK 0
 850 GO SUB 690
 860 RESTORE 930
 870 FOR f=14 TO 17
 880 READ k
 890 FOR c=0 TO 31
 900 PRINT  PAPER k; BRIGHT 1;AT f,c;" "
 910 NEXT c: NEXT f: BRIGHT 0
 920 GO TO 940
 930 DATA 2,6,4,5
 940 RESTORE 950
 950 DATA .1,7,.3,12,.1,7,.3,12,.1,16,.6,19,.12,16,.12,19,.12,19,.12,19,.25,17,.
12,14,.6,12
 960 DATA  .4,-60,.4,19,.4,18,.4,17,.4,16,.2,14,.15,12,.1,11,.1,-60,.15,9,.3,7
 970 FOR b=1 TO 24
 980 READ t,n: BEEP t,n
 990 NEXT b
1000 PRINT AT 21,2;"(C)"
1010 LET p$="J.M. MAYORAL SERRANO": LET yy=160: LET xs=1: LET ys=2: INK 0: GO SU
B 690
1020 PAUSE 150: RUN 40
1030 CLEAR 32255: LOAD "LIT"CODE
1040 GO TO 1070
1050 CLS : LET yy=60: LET xs=1: LET ys=2: LET p$="Vuelvo a REPETIRTELO!!": GO SU
B 690
1060 PAUSE 100: CLS : GO TO 400
1070 BRIGHT 0: BORDER 7: PAPER 7: INK 1: CLS
1080 LET yy=80: LET ys=3: LET xs=2: LET p$="Para la cinta": GO SUB 690
1090 PAUSE 200
1100 CLS : BORDER 0: PAPER 0: INK 6: BRIGHT 1: CLS
1110 LET P$="INTRODUCE UN NUMERO   (0-9999)": LET YY=160: LET XS=1: LET YS=2: IN
K 4: GO SUB 690
1120 INPUT  LINE M$
1130 IF LEN M$>4 THEN  GO TO 1120
1140 RANDOMIZE INT (RND*VAL M$)
1150 INK 6: CLS
1160 LET p$="IMPORTANTE": LET yy=10: LET xs=3: LET ys=5: GO SUB 690: BRIGHT 0:
1170 LET yy=70: LET ys=2: LET xs=1: LET p$="En el momento de cantar": INK 7: GO
SUB 690
1180 LET yy=90: LET p$="LINEA o BINGO": GO SUB 690
1190 LET yy=110: LET p$="MANTENER pulsada la letra ""B""": GO SUB 690
1200 LET yy=130: LET p$="hasta que aparezca un menu": GO SUB 690
1210 LET yy=170: LET xs=2: LET ys=2: LET p$="VALE??": GO SUB 690
1220 PAUSE 300: GO TO 790
```